Scaling Agile Frameworks and Global Software Development Risk: Supplementary Data

Companion to manuscript:

Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study

5th May, 2020

Lero Technical Report No. TR_2020_03

Sarah Beecham, Tony Clear, Ramesh Lal, John Noll

contact: sarah.beecham@lero.ie

in collaboration with

AUCKLAND UNIVERSITY OF TECHNOLOGY, AUCKLAND, NEW ZEALAND

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Science Foundation Ireland Research Centre for Software (www.lero.ie)

Contents

List of Figures

List of Tables

1 Preamble

This technical report documents the protocol and processes we followed in *collecting* and *analysing* data in our study that asks: "*Do practices in scaling agile frameworks help eliminate or mitigate global software development risks?*". The main purpose of this report is to present the raw *results* of our analyses, that due to their complexity and involvement of two separate case studies and four researchers were too lengthy to include in the associated paper [2].

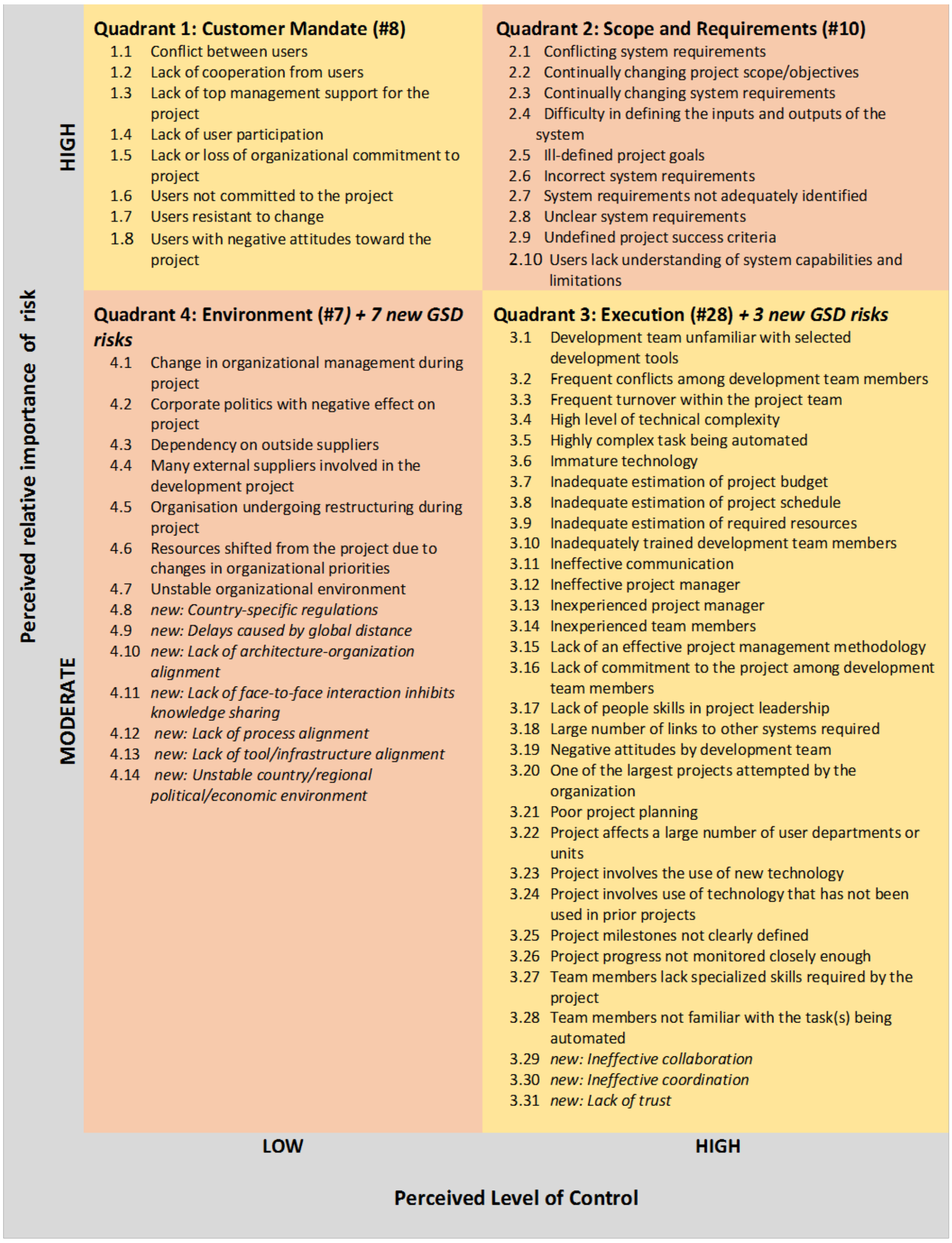
1.1 Study Context

The study observes how global software development companies need to coordinate activities of multiple agile development teams, who must cooperate to produce large software products. Such software intensive organizations are turning to scaling agile software development frameworks to support their endeavours, yet, despite their growing adoption, little is known about how effective these scaling agile frameworks are in mitigating risk, especially in global software development (GSD), where project failure is a known problem.

1.2 Material Included

This technical report contains:

1. Details of how we developed the *GSD Risk Catalog* subsequently used to assess GSD risks experienced in two case studies, see Fig. 1, Table 1 and Table 2 in Section 2.
2. Our *Theoretical Scaling Agile Risk Mitigation Model*, which measures the degree to which two scaling agile frameworks–Disciplined Agile Delivery (DAD) and the Scaled Agile Framework(SAFe)–address software project risks in GSD according to our GSD Risk Catalog. See Table 3 and Table 4 in Section 3.2 and Section 3.3.
3. Our *Empirical Investigation*, in which we test the Theoretical Scaling Agile risk mitigation model using real-world data from two case studies (Case study A who are implementing DAD, the Case study B who are implementing SAFe). See Table 5 and Table 6 in Section 4.
4. Our empirical study involves a *data collection* from a variety of sources, to include observations, examination of documentation, surveys [10] and interviews. Our submitted paper contains many quotes to illustrate the results of our analyses [2] most of which came from interviews. The protocol for Case A (DAD) is presented in the Appendix of this document Section A. The Case B interview protocol can be found in a separate technical report [3] - url: https://www.lero.ie/sites/default/files/Lero_TR_2017_02_Beecham_Noll_Razzak-Lean%20Global%20Project%20Interview%20Protocol.pdf.


Perceived relative importance of risk
HIGH
MODERATE
LOW
HIGH
Perceived Level of Control

Quadrant 1: Customer Mandate (#8)

1.1 Conflict between users
1.2 Lack of cooperation from users
1.3 Lack of top management support for the project
1.4 Lack of user participation
1.5 Lack or loss of organizational commitment to project
1.6 Users not committed to the project
1.7 Users resistant to change
1.8 Users with negative attitudes toward the project

Quadrant 2: Scope and Requirements (#10)

2.1 Conflicting system requirements
2.2 Continually changing project scope/objectives
2.3 Continually changing system requirements
2.4 Difficulty in defining the inputs and outputs of the system
2.5 Ill-defined project goals
2.6 Incorrect system requirements
2.7 System requirements not adequately identified
2.8 Unclear system requirements
2.9 Undefined project success criteria
2.10 Users lack understanding of system capabilities and limitations

Quadrant 4: Environment (#7) *+ 7 new GSD risks*

4.1 Change in organizational management during project
4.2 Corporate politics with negative effect on project
4.3 Dependency on outside suppliers
4.4 Many external suppliers involved in the development project
4.5 Organisation undergoing restructuring during project
4.6 Resources shifted from the project due to changes in organizational priorities
4.7 Unstable organizational environment
4.8 *new: Country-specific regulations*
4.9 *new: Delays caused by global distance*
4.10 *new: Lack of architecture-organization alignment*
4.11 *new: Lack of face-to-face interaction inhibits knowledge sharing*
4.12 *new: Lack of process alignment*
4.13 *new: Lack of tool/infrastructure alignment*
4.14 *new: Unstable country/regional political/economic environment*

Quadrant 3: Execution (#28) *+ 3 new GSD risks*

3.1 Development team unfamiliar with selected development tools
3.2 Frequent conflicts among development team members
3.3 Frequent turnover within the project team
3.4 High level of technical complexity
3.5 Highly complex task being automated
3.6 Immature technology
3.7 Inadequate estimation of project budget
3.8 Inadequate estimation of project schedule
3.9 Inadequate estimation of required resources
3.10 Inadequately trained development team members
3.11 Ineffective communication
3.12 Ineffective project manager
3.13 Inexperienced project manager
3.14 Inexperienced team members
3.15 Lack of an effective project management methodology
3.16 Lack of commitment to the project among development team members
3.17 Lack of people skills in project leadership
3.18 Large number of links to other systems required
3.19 Negative attitudes by development team
3.20 One of the largest projects attempted by the organization
3.21 Poor project planning
3.22 Project affects a large number of user departments or units
3.23 Project involves the use of new technology
3.24 Project involves use of technology that has not been used in prior projects
3.25 Project milestones not clearly defined
3.26 Project progress not monitored closely enough
3.27 Team members lack specialized skills required by the project
3.28 Team members not familiar with the task(s) being automated
3.29 *new: Ineffective collaboration*
3.30 *new: Ineffective coordination*
3.31 *new: Lack of trust*

Figure 1: GSD Risk Catalog derived from Wallace and Keil [12] and Verner et al. [11]

2 Risk Framework Mappings

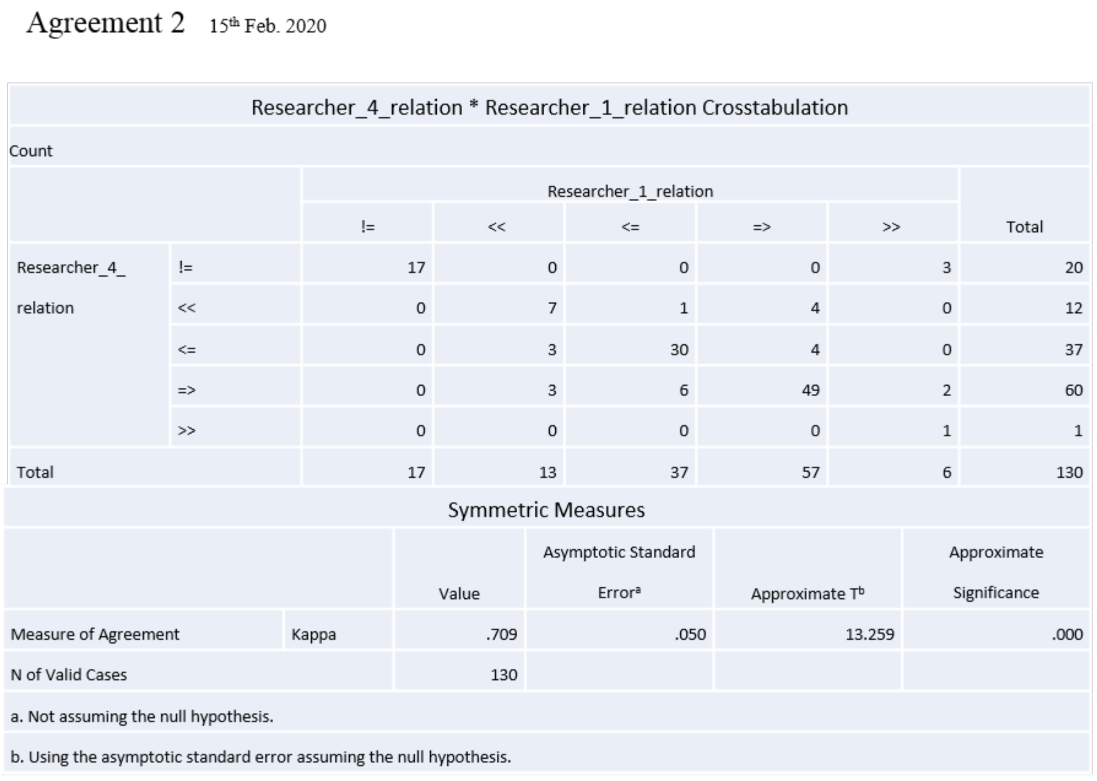
2.1 Persson to Wallace and Keil

In order to update the recognised set of 53 software development risk factors according to Wallace and Keil [12] with GSD risks we first looked at Persson et al's [9] empirical evaluation of risks observed in distributed settings. Rather than blindly adding the new risks to the Wallace and Keil risks, we decided to look at how the two sets of risks could be merged, or where new risks may need to be defined in this new context.

Our mapping followed a series of steps, that involved all four researchers.

- Step 1. Three researchers (1, 2 and 3) all independently mapped the Perrson risks to Wallace and Keil.
- Step 2. Researchers 2 and 3 met to discuss differences and reached a consensus.
- Step 3. Researcher 1, compared her mappings with those of researchers 2 and 3, in which initial agreement was fairly high. Since there were some disagreements, Researcher 4 acted as a moderator.
- Step 4. Researcher 4, in order to moderate, conducted an independent mapping, and identified many different types of relationships [1] between the two sets of risk. A series of mappings were questioned, and new mappings introduced.
- Step 5. Researcher 1 went through all her mappings again, checked the new mappings introduced by Researcher 4, and added her view of the new relationships. According to Cohen's Kappa results (see Fig. 2), showing the .709 measure of agreement with 130 cases, which according to McHugh [7] represents a substantial agreement (falling into the k band of values of between 0.61 to 0.80).
- Step 6. Researchers 1 and 4 compared results and discussed differences in risk relationships and mappings, until a consensus was reached.
- Step 7. Finally, the mappings were checked by Researcher 2, who approved new mappings, and defended where possible a few of the mappings that researchers 1 and 4 queried.

[1]There are six types of relationship between Wallace & Keil Relationship key: (1) Subset « *W&K is an example of a Persson risk*; (2) Subset » *Persson is an example of a W&K risk*; (3) != *not equivalent, no direct relationship*; (4) <= *Persson risk led to or triggers a W&K risk*; (5) => *W&K risk led to or triggers Persson Risk*; (6) <=> *Bi-directional influence; W&K risk can be triggered by Persson, or vice versa*



Agreement 2 15th Feb. 2020

Researcher_4_relation * Researcher_1_relation Crosstabulation

Count

		Researcher_1_relation					
		!=	<<	<=	=>	>>	Total
Researcher_4_ relation	!=	17	0	0	0	3	20
	<<	0	7	1	4	0	12
	<=	0	3	30	4	0	37
	=>	0	3	6	49	2	60
	>>	0	0	0	0	1	1
Total		17	13	37	57	6	130

Symmetric Measures

		Value	Asymptotic Standard Error[a]	Approximate T[b]	Approximate Significance
Measure of Agreement	Kappa	.709	.050	13.259	.000
N of Valid Cases		130			

a. Not assuming the null hypothesis.

b. Using the asymptotic standard error assuming the null hypothesis.

Figure 2: Cohen Kappa agreement scores for Researchers 1 and 4 for Persson et al [9] mapping to Wallace and Keil[12] risks

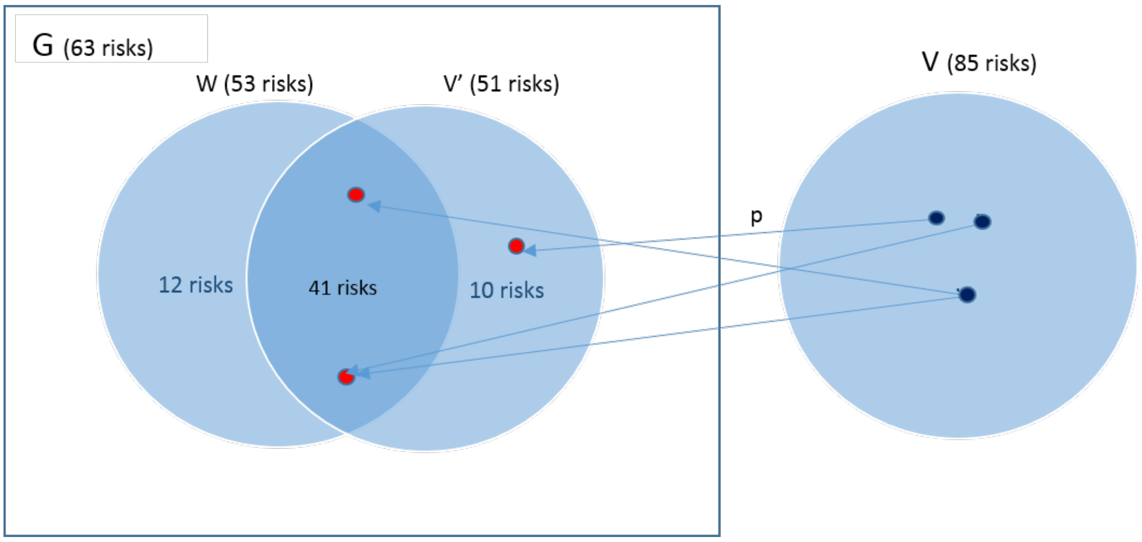
The final, agreed mapping of Persson et al's risks to Wallace and Keil's risks is presented in Table 1.

Having looked further into other possible mappings we decided that by applying Verner et al's[11] 85 risks (that included all the Persson et al's risks we mapped in this section) we would capture more of a cross section of GSD risks, as Verner et al's SLR draws on 24 studies on risk in GSD. The method followed for the Verner et al mapping to Wallace and Keil is as follows:

First, two researchers independently compared *each* risk identified by Verner and colleagues to *each* risk in Wallace and Keil's risk catalog. If the risk from Wallace and Keil was equivalent to, or would be a consequence of, a risk from Verner et al., we created a correspondence between the two. Mapping the two sets of risks was not straightforward, since some risks in Verner and colleagues' catalog are expressed at a high level, or as a combination of risks; in such cases, the risk from Verner et al was mapped to multiple Wallace and Keil risks. We marked those risks in Verner and colleagues' catalog that did not correspond to a Wallace and Keil risk, or were incompletely captured by a set of Wallace and Keil risks, for later consideration.

Second, we reviewed the independent mappings to resolve disagreements between the two researchers, to create a single unified mapping.

Third, we coalesced unmapped or partially mapped risks into a small set of new risks. Then, we repeated the first two steps on these new risks, to create a candidate combined GSD Risk Catalog of 63 risks. Next, each researcher independently reviewed the candidate GSD Risk Catalog to double-check that the mappings made sense, and placed the new risks into the relevant quadrant.



In the diagram above, let G be the set of risks in our GSD Risk Catalog, V be the set of risks identified by Verner and colleagues [11], and M be the set of risks catalogued by Wallace and Keil [12]. Then,

$$p : V \mapsto W$$

where p is the mapping process we employed to relate risks in Verner et al. to risks in Wallace and Keil's catalog, and V' is the subset of risks in the GSD Risk Catalog to which one or more risks in Verner et al. are mapped (i.e. the co-domain of p). The subset of G comprising ten new risks created to account for risks from Verner et al. that had no, or incomplete, mapping to risks in Wallace and Keil's catalog is the relative complement of W with respect to V' ($V' \setminus W$).

Figure 3: GSD Risk Catalog creation based on Verner et al. [11] and Wallace & Keil [12]

Finally, we reviewed these checks to ensure agreement on the final GSD Risk Catalog. Table 2 shows an extract of the results as published in [2]. However in this Technical Report we provide the full mapping. The numbers of practices mapped (or not) are summarised in the Venn diagram in Fig. 3. As is shown here, we added ten risks to Wallace and Keil's inventory, derived from the Verner and colleague's set of risks that do not map, or only partly mapped, to risks of Wallace and Keil. The remaining 75 risks in Verner et al. corresponded to one or more of 41 of Wallace and Keil's risks. Finally, there were 12 risks from Wallace and Keil that did not have a corresponding Verner et al. risk.

The full list of 63 risks in our derived GSD Risk Catalog is presented in Fig. 1.

Table 1: Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
Quadrant 1: Customer Mandate		
Conflict between users		
⇐	Stakeholder Relations	Mutual Trust
Lack of cooperation from users		
⇐	Stakeholder Relations	Relationship Bulding

Continued on next page.

Table 1 (continued): Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
Lack of top management support for the project		
$\subset$	Stakeholder Relations	Stakeholder Commitment
Lack of user participation		
$\Leftarrow$	Stakeholder Relations	Stakeholder Commitment
Lack or loss of organizational commitment to the project		
$\subset$	Stakeholder Relations	Stakeholder Commitment
Users not committed to the project		
$\subset$	Stakeholder Relations	Stakeholder Commitment
Users resistant to change		
$\Leftarrow$	Stakeholder Relations	Relationship Bulding
Users with negative attitudes toward the project		
$\Leftarrow$	Stakeholder Relations	Relationship Bulding
Quadrant 2: Scope and Requirements		
Conflicting system requirements		
$\Leftarrow$	Task Distribution	Task Coupling
$\Rightarrow$	Task Distribution	Task Uncertainty
$\Rightarrow$	Task Distribution	Task Equivocality
Continually changing project scope/objectives		
$\Leftarrow$	Task Distribution	Task Coupling
$\Rightarrow$	Task Distribution	Task Uncertainty
$\Rightarrow$	Task Distribution	Task Equivocality
Continually changing system requirements		
$\Leftarrow$	Task Distribution	Task Coupling
$\Rightarrow$	Task Distribution	Task Uncertainty
$\Rightarrow$	Task Distribution	Task Equivocality
Difficulty in defining the inputs and outputs of the system		
$\subset$	Knowledge Management	Knowledge Creation
$\Leftarrow$	Knowledge Management	Knowledge Capture
$\Leftarrow$	Knowledge Management	Knowledge Integration
Ill-defined project goals		
$\Leftarrow$	Geographic Distribution	Goal Distribution
$\Rightarrow$	Task Distribution	Task Uncertainty
$\Rightarrow$	Task Distribution	Task Equivocality
Incorrect system requirements		
$\Rightarrow$	Task Distribution	Task Uncertainty
$\Rightarrow$	Task Distribution	Task Equivocality
System requirements not adequately identified		
$\Rightarrow$	Task Distribution	Task Equivocality

Continued on next page.

Table 1 (continued): Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
Unclear system requirements		
⇒	Task Distribution	Task Equivocality
Undefined project success criteria		
⇒	Task Distribution	Task Equivocality
Users lack understanding of system capabilities and limitations		
⇐	Stakeholder Relations	Relationship Building
⇐	Knowledge Management	Knowledge Integration
⇒	Knowledge Management	Knowledge Creation
⇒	Knowledge Management	Knowledge Capture
	Quadrant 3: Execution	
Development team unfamiliar with selected development tools		
⇐	Knowledge Management	Knowledge Integration
⇒	Knowledge Management	Knowledge Creation
⇒	Knowledge Management	Knowledge Capture
⊂	Technology Setup	Tool Compatibility
Frequent conflicts among development team members		
⇐	Task Distribution	Task Equivocality
⇐	Stakeholder Relations	Mutual Trust
⇐	Stakeholder Relations	Relationship Building
Frequent turnover within the project team		
⇒	Task Distribution	Task Equivocality
⇐	Stakeholder Relations	Stakeholder Commitment
	Stakeholder Relations	Mutual Trust
	Stakeholder Relations	Relationship Building
High level of technical complexity		
⇒	Task Distribution	Task Uncertainty
⇒	Task Distribution	Task Equivocality
⇒	Task Distribution	Task Coupling
Highly complex task being automated		
⇒	Task Distribution	Task Uncertainty
⇒	Task Distribution	Task Equivocality
⇒	Task Distribution	Task Coupling
Immature technology		
⊃	Technology Setup	Network Capability
⇒	Task Distribution	Task Uncertainty
⇒	Task Distribution	Task Equivocality
⇐	Technology Setup	Configuration Management
⇒	Technology Setup	Tool Compatibility
Inadequate estimation of project budget		
⇐	Task Distribution	Task Uncertainty

Continued on next page.

Table 1 (continued): Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
$\Leftarrow$	Task Distribution	Task Equivocality
Inadequate estimation of project schedule		
$\Leftarrow$	Task Distribution	Task Uncertainty
$\Leftarrow$	Task Distribution	Task Equivocality
Inadequate estimation of required resources		
$\Leftarrow$	Task Distribution	Task Uncertainty
Inadequately trained development team members		
	Knowledge Management	Knowledge Capture
$\Rightarrow$	Collaboration Structure	Collaboration Capability
Ineffective communication		
$\subset$	Collaboration Structure	Coordination Capability
$\Leftarrow$	Cultural Distribution	Language Barriers
$\Leftarrow$	Communication Infrastructure	Interaction Media
$\Leftarrow$	Communication Infrastructure	Teleconference Managaement
$\supset$	Communication Infrastructure	Personal communication
$\Leftarrow$	Technology Setup	Network Capability
$\Leftarrow$	Technology Setup	Tool Compatibility
Ineffective project manager		
$\Leftarrow$	Collaboration Structure	Collaboration Capability
$\Leftarrow$	Collaboration Structure	Collaboration Mechanisms
Inexperienced project manager		
$\Rightarrow$	Collaboration Structure	Collaboration Capability
Inexperienced team members		
$\Rightarrow$	Knowledge Management	Knowledge Creation
$\Rightarrow$	Collaboration Structure	Collaboration Capability
Lack of an effective project management methodology		
$\Leftarrow$	Collaboration Structure	Process Alignment
$\Rightarrow$	Collaboration Structure	Collaboration Capability
$\Rightarrow$	Collaboration Structure	Coordination Mechanisms
Lack of commitment to the project among development team members		
$\subset$	Stakeholder Relations	Stakeholder Commitment
Lack of people skills in project leadership		
$\Leftarrow$	Cultural Distribution	Cultural Bias
$\Rightarrow$	Stakeholder Relations	Mutual Trust
$\Rightarrow$	Stakeholder Relations	Relationship Building
$\Rightarrow$	Collaboration Structure	Collaboration Capability
Large number of links to other systems required		
$\Rightarrow$	Task Distribution	Task Coupling
Negative attitudes by development team		

Continued on next page.

Table 1 (continued): Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
⇐	Stakeholder Relations	Relationship Building
One of the largest projects attempted by the organization		
⇒	Task Distribution	Task Equivocality
⇒	Task Distribution	Task Coupling
⇒	Collaboration Structure	Collaboration Capability
Poor project planning		
	Collaboration Structure	Collaboration Capability
⇒	Collaboration Structure	Coordination Mechanisms
Project affects a large number of user departments or units		
⇒	Task Distribution	Task Coupling
Project involves the use of new technology		
⇒	Technology Setup	Tool Compatibility
Project involves use of technology that has not been used in prior projects		
⇒	Technology Setup	Tool Compatibility
Project milestones not clearly defined		
⇐	Collaboration Structure	Collaboration Capability
⇐	Collaboration Structure	Collaboration Mechanisms
Project progress not monitored closely enough		
⇐	Collaboration Structure	Collaboration Mechanisms
Team members lack specialized skills required by the project		
⇐	Knowledge Management	Knowledge Integration
⇒	Collaboration Structure	Collaboration Capability
Team members not familiar with the task(s) being automated		
⇒	Task Distribution	Task Uncertainty
⇒	Task Distribution	Task Equivocality
⇐	Knowledge Management	Knowledge Capture
⇐	Knowledge Management	Knowledge Integration
⇐	Knowledge Management	Knowledge Creation
Quadrant 4: Environment		
Change in organizational management during the project		
⇒	Collaboration Structure	Collaboration Capability
⊂	Stakeholder Relations	Stakeholder Commitment
Corporate politics with negative effect on project		
⇒	Stakeholder Relations	Relationship Building
⇒	Stakeholder Relations	Stakeholder Commitment
Dependency on outside suppliers		
⇒	Stakeholder Relations	Relationship Building
⇒	Task Distribution	Task Coupling

Continued on next page.

Table 1 (continued): Mapping of risks from Persson et al. [9] to Wallace and Keil [12] risks.

Rel.	Persson risk area	Persson risk factor
Inadequate estimation of required resources		
⇐	Task Distribution	Task Equivocality
Many external suppliers involved in the development project		
⇒	Stakeholder Relations	Relationship Building
⇒	Task Distribution	Task Coupling
Organization undergoing restructuring during the project		
⇒	Stakeholder Relations	Stakeholder Commitment
⇒	Stakeholder Relations	Relationship Building
Resources shifted from the project due to changes in organizational priorities		
⇒	Task Distribution	Task Uncertainty
	Stakeholder Relations	Stakeholder Commitment
Unstable organizational environment		
⇒	Stakeholder Relations	Stakeholder Commitment
⇒	Stakeholder Relations	Relationship Building

The reason we include the Perrson et al mapping to Wallace and Keil in this Technical Report, is to show how robust Wallace and Keil's set of risks are, in that almost all of the Persson et al GSD risks can be mapped to the Wallace and Keil framework.

See next section for how we augment the Wallace and Keil (53) risks with 10 new GSD risks. The final catalog of 63 risks derived from the broader mapping is described in Fig. 1 as used in our paper [2].

2.2 Verner to GSD Risk Catalog

Table 2: Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Quadrant 1: Customer Mandate
Conflict between users
National, organizational, and cultural differences of participants can cause problems like rework, loss of data, confusions, etc.
Lack of cooperation from users
National, organizational, and cultural differences of participants can cause problems like rework, loss of data, confusions, etc.
Lack of user participation
National, organizational, and cultural differences of participants can cause problems like rework, loss of data, confusions, etc. Lack of collaboration for RE between distributed stakeholders happens due to differences in culture, language distance and processes Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Requirements information not properly shared with distributed stakeholders affecting their interaction
Lack or loss of organizational commitment to the project
Stakeholders are less likely to commit to the project organization and its task when cultural differences and lack of face-to-face interaction makes it difficult to establish a clear project identity
Users not committed to the project
Stakeholders are less likely to commit to the project organization and its task when cultural differences and lack of face-to-face interaction makes it difficult to establish a clear project identity
Users with negative attitudes toward the project
Lack of informal communication negatively impacts relationship building
Quadrant 2: Scope and Requirements
Conflicting system requirements
Lack of a common understanding of requirements leads to problems in system functionality Lack of collaboration for RE between distributed stakeholders happens due to differences in culture, language distance and processes Requirements information not properly shared with distributed stakeholders affecting their interaction
Continually changing project scope/objectives
Risk management including issues related to coordination, problem resolution problem resolution evolving requirements, knowledge sharing, and risk identification Task uncertainty represents lack of information needed to develop the software, and it can result in slow change coordination and process and relational conflict High organizational complexity, scheduling, task assignment and cost estimation become more problematic in distributed environments as a result of volatile requirements diversity and lack of informal communication
Continually changing system requirements
Risk management including issues related to coordination, problem resolution problem resolution evolving requirements, knowledge sharing, and risk identification High organizational complexity, scheduling, task assignment and cost estimation become more problematic in distributed environments as a result of volatile requirements diversity and lack of informal communication Requirements change management issues can be a problem especially if there are no defined organizational policies
Difficulty in defining the inputs and outputs of the system
Lack of effective communication causes problems with knowledge management; participants may lack information about tasks, purpose, and their own contribution overall Lack of appropriate information flow throughout the project causes problems with knowledge management
Ill-defined project goals
Poor communication bandwidth for agile development causes problems with communication and knowledge management

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Goal distribution can lead to conflicts related to task interpretation, process principles, and problem resolution approaches and can result in site wars and low performance. Goal distribution is more likely in GSD because of faulty transfer of information and a focus on own site performance
Lack of coordination causes problems such as unclear lines of communication and poor handling of deadlines and milestones
Problems with tracking and control can lead to unawareness of real project progress
Task uncertainty represents lack of information needed to develop the software, and it can result in slow change coordination and process and relational conflict
Lack of common stakeholder goals due to problems of communication and lack of common understanding of requirements
Incorrect system requirements
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Problems caused because team members do not share equal knowledge of the domain
Lack of a common understanding of requirements leads to problems in system functionality
Lack of collaboration for RE between distributed stakeholders happens due to differences in culture, language distance and processes
System requirements not adequately identified
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
A lack of suitable tools or methodologies available for requirements elicitation may lead to problems in obtaining the real requirements
Requirements information not properly shared with distributed stakeholders affecting their interaction
Unclear system requirements
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Cultural diversity between development sites or teams can cause misunderstandings
Lack of a common understanding of requirements leads to problems in system functionality
Lack of collaboration for RE between distributed stakeholders happens due to differences in culture, language distance and processes
Requirements information not properly shared with distributed stakeholders affecting their interaction
Undefined project success criteria
Spatial distribution complicates project manager’s ability to monitor participants and progress, increases travel budgets, limits face-to-face interaction, and weakens social relations
Goal distribution can lead to conflicts related to task interpretation, process principles, and problem resolution approaches and can result in site wars and low performance. Goal distribution is more likely in GSD because of faulty transfer of information and a focus on own site performance
consider adding two other risks: Problems with tracking and control can lead to unawareness of real project progress

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Lack of effective communication causes problems with knowledge management; participants may lack information about tasks, purpose, and their own contribution overall
Users lack understanding of system capabilities and limitations
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Limited face-to-face meetings caused by geographic distance impact trust, decision quality, creativity, and general management; knowledge creation is limited within organization. This may lead to problems in creating collaboration know-how and domain knowledge
Quadrant 3: Execution
Development team unfamiliar with selected development tools
Working with different CM tools can cause slow and unreliable sites, lack of awareness of product changes, and problems with bug fixes between sites
Working in different SCM environments leads to maintenance requests being handled at several levels in the project
Selection of inappropriate information and communication technology causes problems such as unreliable networks which may lead to frustration and low efficiency, limit exchange of sensitive information, or even cause production to stop
Lack of training in communication and collaboration tools causes problems with communication
Frequent conflicts among development team members
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Cultural bias may lead to erroneous decisions and insecurity about other participants' qualifications and it can have a devastating impact on communication and collaboration efforts. Cultural bias occurs when project participants consider their norms and values as universal and neglect to reflect onto what extent values, norms, and biases are founded in their own cultural background
Cultural diversity between development sites or teams can cause misunderstandings
Differences in work culture may render difficulties when sites are different in terms of team behaviour, balancing of collectivism and individualism, perception of authority and hierarchy, planning, punctuality, and organizational culture. This may lead to decreased conflict handling capabilities and lower efficiency or even paralyse the project
Not tailoring organizational structures to reduce delays in problem resolution causes difficulties and can result in site wars and reduce project cohesion
Task uncertainty represents lack of information needed to develop the software, and it can result in slow change coordination and process and relational conflict
Problems with people management, conflict resolution, and staff turnover are caused by a lack of control
Frequent turnover within the project team
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Problems with people management, conflict resolution, and staff turnover are caused by a lack of control
Immature technology
Selection of inappropriate information and communication technology causes problems such as unreliable networks which may lead to frustration and low efficiency, limit exchange of sensitive information, or even cause production to stop
Problems can be caused by the lack of deployment of a configuration management system

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Selection of inappropriate information and communication technology causes problems such as unreliable networks which may lead to frustration and low efficiency, limit exchange of sensitive information, or even cause production to stop
Inadequate estimation of project budget
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Spatial distribution complicates project manager's ability to monitor participants and progress, increases travel budgets, limits face-to-face interaction, and weakens social relations High organizational complexity, scheduling, task assignment and cost estimation become more problematic in distributed environments as a result of volatile requirements diversity and lack of informal communication
Inadequate estimation of project schedule
High organizational complexity, scheduling, task assignment and cost estimation become more problematic in distributed environments as a result of volatile requirements diversity and lack of informal communication Poor schedule management
Inadequate estimation of required resources
Spatial distribution complicates project manager's ability to monitor participants and progress, increases travel budgets, limits face-to-face interaction, and weakens social relations High organizational complexity, scheduling, task assignment and cost estimation become more problematic in distributed environments as a result of volatile requirements diversity and lack of informal communication
Inadequately trained development team members
Lack of training in communication and collaboration tools causes problems with communication
Ineffective communication
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Lack of synchronous communication in agile development causes problems Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts Poor communication bandwidth for agile development causes problems with communication and knowledge management Large teams involved with agile development can cause problems related to communication and coordination Working in different SCM environments leads to maintenance requests being handled at several levels in the project Language differences between development sites can result in misinterpretations and unconveyed information Cultural bias may lead to erroneous decisions and insecurity about other participants' qualifications and it can have a devastating impact on communication and collaboration efforts. Cultural bias occurs when project participants consider their norms and values as universal and neglect to reflect onto what extent values, norms, and biases are founded in their own cultural background Cultural diversity between development sites or teams can cause misunderstandings Lack of training in communication and collaboration tools causes problems with communication

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Limited possibility for informal communication due to dispersion of sites (i.e., lack of spontaneous communication) causes problems with social integration of teams
Lack of effective communication causes problems with knowledge management; participants may lack information about tasks, purpose, and their own contribution overall
Inability to communicate in real time) causes collaboration problems
It may be difficult to establish effective coordination mechanisms in projects and overcome challenges such as little face-to-face interaction, problematic task coupling, different time zones, local holidays, weak social networks, and unclear communication lines
Problems caused by poor collaboration and communication infrastructure
Use of interaction media may cause problems such as jumbled messages; mix-ups and loss of contextual information. Such problems, may lead to confusion and misunderstandings among participants and lower morale
Selection of inappropriate information and communication technology causes problems such as unreliable networks which may lead to frustration and low efficiency, limit exchange of sensitive information, or even cause production to stop
Without communication protocols effective communication can be impeded
Language and cultural barriers cause problems between client and vendor
A communication gap between client and vendor may lead to misunderstandings
Lack of informal communication negatively impacts relationship building
Stakeholders located in different time zones can lead to problems in communicating
Ineffective project manager
Lack of training in communication and collaboration tools causes problems with communication
Poor quality management
Choosing a vendor with a lack of project management skills can result in difficulties
Inexperienced project manager
Choosing a vendor with a lack of project management skills can result in difficulties
Inexperienced team members
Lack of training in communication and collaboration tools causes problems with communication
Lack of an effective project management methodology
Temporal and physical distribution increases complexity of planning and coordination activities, makes multisite virtual meetings hard to plan, causes unproductive waits, delays feedback, and complicates simple things
Poor quality management
Risk management including issues related to coordination, problem resolution problem resolution evolving requirements, knowledge sharing, and risk identification
Lack of coordination causes problems such as unclear lines of communication and poor handling of deadlines and milestones
Problems with tracking and control can lead to unawareness of real project progress
Poor control results in lack of effective scope and change management
Disorganized task allocation leads to some work being done twice and other work omitted
Lack of control results in no transparency or visibility of project status to all sites involved in project
Poor schedule management
Choosing a vendor with a lack of project management skills can result in difficulties
A vendor with poor contract management can cause problems for the client such as lack of integrity in obligations, commitments and behaviou

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Problems caused by asymmetry in processes, policies and standards
Lack of commitment to the project among development team members
Stakeholders are less likely to commit to the project organization and its task when cultural differences and lack of face-to-face interaction makes it difficult to establish a clear project identity
Lack of people skills in project leadership
Problems with people management, conflict resolution, and staff, turnover are caused by a lack of control
Negative attitudes by development team
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts Cultural bias may lead to erroneous decisions and insecurity about other participants qualifications and it can have a devastating impact on communication and collaboration efforts. Cultural bias occurs when project participants consider their norms and values as universal and neglect to reflect onto what extent values, norms, and biases are founded in their own cultural background
new: Lack of architecture-organization alignment
Lack of well-defined modules causes problems with progressive integration High task coupling between task segments increases the need for inter-site communication, coordination, and integration, and it can lead to lower level of performance as well as increase the number of failures
new: Lack of trust
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts Mutual trust is important but hard to obtain and lack of trust causes problems. This can be due to lack of face-to-face interaction, cultural differences, and weak social relations Fear about the future of jobs and roles, erodes trust Limited face-to-face meetings caused by geographic distance impact trust, decision quality, creativity, and general management; knowledge creation is limited within organization. This may lead to problems in creating collaboration know-how and domain knowledge Trust among stakeholders necessary to achieve innovation, flexibility, cooperation, and efficiency in distributed environment. Since often a short life span, important to achieve mutual trust rapidly, but if trust is misplaced, entire organization may suffer A vendor with poor relationship management can result in problems such as lack of trust
Poor project planning
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Lack of project planning causes problems Poor identification of roles and responsibilities Lack of detailed planning causes problems with task allocation
Project affects a large number of user departments or units
Organizational challenges caused by GSD beyond distance and cultural differences, e.g., if the client organization has a large number of stakeholders, and /or the vendor organization has a number of sites Large teams involved with agile development can cause problems related to communication and coordination

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Project involves use of technology that has not been used in prior projects
Working with different CM tools can cause slow and unreliable sites, lack of awareness of product changes, and problems with bug fixes between sites Working in different SCM environments leads to maintenance requests being handled at several levels in the project Choosing a vendor with a lack of technical capability can result in problems
Project milestones not clearly defined
Lack of coordination causes problems such as unclear lines of communication and poor handling of deadlines and milestones
Project progress not monitored closely enough
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Lack of coordination causes problems such as unclear lines of communication and poor handling of deadlines and milestones Problems with tracking and control can lead to unawareness of real project progress Lack of control results in no transparency or visibility of project status to all sites involved in project Poor schedule management Spatial distribution complicates project manager's ability to monitor participants and progress, increases travel budgets, limits face-to-face interaction, and weakens social relations
Team members lack specialized skills required by the project
Lack of training in communication and collaboration tools causes problems with communication Difficulties caused by different knowledge levels or knowledge transfer problems Choosing a vendor with a lack of technical capability can result in problems
Team members not familiar with the task(s) being automated
Difficulties caused by different knowledge levels or knowledge transfer problems Problems caused because team members do not share equal knowledge of the domain Lack of appropriate information flow throughout the project causes problems with knowledge management Task uncertainty represents lack of information needed to develop the software, and it can result in slow change coordination and process and relational conflict
Quadrant 4: Environment
Dependency on outside suppliers
Vendor's strategic inflexibility can result in major disagreements Vendor's poor infrastructure such as infrastructure incompatibility between sites causes problems; selection of appropriate information and communication technology is crucial for project success Lack of protection for intellectual property rights in the vendor country Problems because of differences in legal systems such as jurisdiction, patents, and International laws Vendor country instability such as political instability, corruption, peace problems, terrorism threats and uncertainty relating to trade and investment can cause project difficulties Vendor's behaves opportunistically Vendor has previous delays in delivery so may be unreliable Vendor incompatibility with client causes problems Choosing a vendor with a lack of project management skills can result in difficulties

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Choosing a vendor with a lack of technical capability can result in problems A vendor with poor contract management can cause problems for the client such as lack of integrity in obligations, commitments and behaviou A vendor with poor quality of service and systems/processes can result in problems Hidden vendor costs can be expensive Choosing a vendor with a lack of control over a project can result in problems such as cost and schedule overruns
new: Country-specific regulations
Lack of protection for intellectual property rights in the vendor country Problems because of differences in legal systems such as jurisdiction, patents, and International laws
new: Delays caused by global distance
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Configuration management problems cause dependency, delay and increased time is required to complete maintenance requests Temporal and physical distribution increases complexity of planning and coordination activities, makes multisite virtual meetings hard to plan, causes unproductive waits, delays feedback, and complicates simple things Inability to communicate in real time) causes collaboration problems; Stakeholders located in different time zones can lead to problems in communicating Not tailoring organizational structures to reduce delays in problem resolution causes difficulties and can result in site wars and reduce project cohesion Choosing a vendor with a lack of control over a project can result in problems such as cost and schedule overruns; and Poor schedule management
new: Ineffective collaboration
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed Lack of tool support for agile development causes problems with agile practices Cultural bias may lead to erroneous decisions and insecurity about other participants' qualifications and it can have a devastating impact on communication and collaboration efforts. Cultural bias occurs when project participants consider their norms and values as universal and neglect to reflect onto what extent values, norms, and biases are founded in their own cultural background Lack of training in communication and collaboration tools causes problems with communication Limited possibility for informal communication due to dispersion of sites (i.e., lack of spontaneous communication) causes problems with social integration of teams Inability to communicate in real time) causes collaboration problems Problems caused by poor collaboration and communication infrastructure Selection of inappropriate information and communication technology causes problems such as unreliable networks which may lead to frustration and low efficiency, limit exchange of sensitive information, or even cause production to stop Language and cultural barriers cause problems between client and vendor Lack of collaboration for RE between distributed stakeholders happens due to differences in culture, language distance and processes
new: Ineffective coordination
Problems caused by asymmetry in processes, policies and standards

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Lack of synchronous communication in agile development causes problems
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Poor communication bandwidth for agile development causes problems with communication and knowledge management
Lack of tool support for agile development causes problems with agile practices
Large teams involved with agile development can cause problems related to communication and coordination
High task coupling between task segments increases the need for inter-site communication, coordination, and integration, and it can lead to lower level of performance as well as increase the number of failures
Temporal and physical distribution increases complexity of planning and coordination activities, makes multisite virtual meetings hard to plan, causes unproductive waits, delays feedback, and complicates simple things
It may be difficult to establish effective coordination mechanisms in projects and overcome challenges such as little face-to-face interaction, problematic task coupling, different time zones, local holidays, weak social networks, and unclear communication lines
Lack of coordination causes problems such as unclear lines of communication and poor handling of deadlines and milestones
Coordination in multisite development becomes more difficult in terms of articulation of work as problems from communication lack of group awareness and complexity of the organization appear and influence the way the work must be structured
new: Lack of face-to-face interaction inhibits knowledge sharing
Application of agile practices causes problems in distributed development because of the degree of interaction between stakeholders and number of face-to-face meetings needed
Collaboration difficulties caused by geographic distance in agile development may cause misunderstandings and conflicts
Poor communication bandwidth for agile development causes problems with communication and knowledge management
Project participants have limited understanding of other project participants' competencies
Lack of team cohesiveness causes problems as some members feel isolated from other team members; participants have limited understanding of other project participants' competencies
Limited face-to-face meetings caused by geographic distance impact trust, decision quality, creativity, and general management; knowledge creation is limited within organization. This may lead to problems in creating collaboration know-how and domain knowledge
Temporal and physical distribution increases complexity of planning and coordination activities, makes multisite virtual meetings hard to plan, causes unproductive waits, delays feedback, and complicates simple things
Coordination in multisite development becomes more difficult in terms of articulation of work as problems from communication lack of group awareness and complexity of the organization appear and influence the way the work must be structured
Spatial distribution complicates project manager's ability to monitor participants and progress, increases travel budgets, limits face-to-face interaction, and weakens social relations
new: Lack of process alignment
Problems caused by asymmetry in processes, policies and standards

Continued on next page.

Table 2 (continued): Mapping of risks from Verner et al. [11] to GSD Risk Catalog risks.

GSD Risk Catalog risk/Mapped risk from Verner et al.
No process alignment, in terms of traditions, development methods, and emphasis on user involvement, will often differentiate between sites, possibly resulting in incompatibility and conflicts
National, organizational, and cultural differences of participants can cause problems like rework, loss of data, confusions, etc.
new: Lack of tool/infrastructure alignment
Tool compatibility may prove a problem; sites are likely to prefer different programming languages, support tools, operating systems, and development tools
Vendor's poor infrastructure such as infrastructure incompatibility between sites causes problems; selection of appropriate information and communication technology is crucial for project success
new: Unstable country/regional political/economic environment
Vendor country instability such as political instability, corruption, peace problems, terrorism threats and uncertainty relating to trade and investment can cause project difficulties

3 Theoretical Mapping

3.1 Theoretical Mapping

Making use of the newly created GSD Risk Catalog described in Section 2, we mapped any practice identified in our Scaling Agile Frameworks, DAD [1] and SAFe [6] to the risk, where they appeared to mitigate risk. This mapping involved three steps:

1. Scaling Agile practice mapped to Risk factors: As part of our ongoing, longitudinal case studies, in previous work we identified sets of practices from SAFe [8], and DAD [4, 5].

   Four researchers, working in pairs, compared each practice in the scaling agile framework, to each of the 63 risks in the GSD Risk Catalog (Fig. 1). To ensure all researchers worked to the same standard, an example of how to 'map' a practice to a risk was shared amongst all researchers.

2. Once the mapping of practices to risks was completed, each risk was rated according to the degree to which the mapped practices eliminated or mitigated the risk, as to whether the practices "definitely" address the risk, address the risk "somewhat", or do "not at all" address the risk. So, if a practice or set of practices unequivocally addressed the risk, we coded the practice as "definitely"; if the practice(s) to some extent contributed to elimination or mitigation, we coded the practice "somewhat"; and when we could not identify a practice that would eliminate or mitigate the risk we coded the risk as "not at all."

3. Inter-rater cross-check (within frameworks): When all possibilities were exhausted, each pair compared their results with each other (DAD researchers compared two sets of independent results, as did the SAFe researchers). Any disagreements were discussed until a consensus was reached.

Output from this theoretical mapping was a theory of risk mitigation according to DAD and SAFe, which is presented in full in Table 3 and Table 4.

3.2 DAD Practice Mapping to GSD Risk Catalog risks

Table 3: DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Quadrant 1: Customer Mandate	
Conflict between users	somewhat
Product manager responsible for features and functionalities, product owner with UX responsibility. Business case, feature funnel, storyboarding, user stories, DOD including acceptance tests, iteration show and tell, beta testing, production environment test, DevOp practices, real time monitoring of the product environment.	
Lack of cooperation from users	somewhat
Product manager, product owner. Business case, feature funnel, storyboarding, DevOp practices, real time monitoring of the product environment by end-users, feedback from end-users.	
Lack of top management support for the project	definitely
Portfolio manager, portfolio planning team. Portfolio planning including feature funnel practices, business case including six monthly presentation to the executives for approval.	
Lack of user participation	somewhat
Product management, vision planning, program management, story boarding. Product manager, proxy user product owner, on-site customer with UX responsibility.	
Lack or loss of organizational commitment to the project	definitely
Product management, portfolio management, program management teams. Product planning, portfolio planning, and program planning. Work item list.	
Users not committed to the project	somewhat
Product manager, product owner, domain expert. Business case, feature funnel, storyboarding, DevOp practices, real time monitoring of the product environment by end-users, feedback from end-users.	
Users resistant to change	definitely
Product management, vision planning, program management, story boarding. Product manager, proxy user product owner, on-site customer with UX responsibility. Both roles require individuals with solid local technical product knowledge.	
Users with negative attitudes toward the project	somewhat
Product manager of features and functionalities, product owner with UX responsibility). Business case, feature funnel, storyboarding, user stories, DOD including acceptance tests, iteration show and tell, beta testing, production environment test, DevOp practices, real time monitoring of the product environment.	
Quadrant 2: Scope and Requirements	
Conflicting system requirements	definitely
Product manager who does market investigation and feedback on potential features and functionalities from potential and current customers. Vision planning, product owner with UX responsibility, for storyboarding, do user research in the field. Engineers in the inception phase minimise risk through spikes. Vision planning, business case, feature funnel.	
Continually changing project scope/objectives	definitely
Program management creates work item list for the entire program. Delivery teams choose user stories for their work item list (project). Program daily huddles. DOD, UAT. Programs are time-boxed, defined, and relatively short duration.	

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Continually changing system requirements	definitely
Product manager who does market investigation and feedback on potential features and functionalities from potential and current customers. Vision planning, product owner with UX responsibility, for storyboarding, do user research in the field. Engineers in the inception phase minimise risk through spikes. Vision planning, business case, feature funnel.	
Difficulty in defining the inputs and outputs of the system	definitely
Program management including Enterprise architects and Product owners. Program work item list, architecture design, story boarding, user stories. Project-level product owners, team architects, software, and QA engineers. Inception phase including spikes to understand users stories. UAT during onstruction phase.	
Ill-defined project goals	definitely
Product owner, Team lead, Architecture owner. Project work item list, DOD, inception phase, construction phase, transition phase.	
Incorrect system requirements	definitely
Product management. Vision planning to identify high level/epic stories. Product owner, story boarding.	
System requirements not adequately identified	definitely
Program management including Product owners, enterprise architects, program manager. Program work item list, story boarding, user stories. Project level product owners, team architects, software and QA engineers. Inception phase project item list, spikes, user stories estimated and prioritised using user story points,	
Unclear system requirements	definitely
Program management including enterprise architects. Program work item list. Architect owner. Spikes.	
Undefined project success criteria	definitely
Portfolio, program management, project management. Portfolio manager, program manager, product owners and enterprise architects. Release plan (roadmap) at portfolio level; program level work item. Iteration show and tell, program daily hurdles, project level DOD and story points.	
Users lack understanding of system capabilities and limitations	somewhat
Product managers, customer success team. Vision planning. Program management does storyboarding for UX.	
Quadrant 3: Execution	
Development team unfamiliar with selected development tools	definitely
DAD delivery teams are empowered to make tool decision	
Frequent conflicts among development team members	definitely
Program management, including program manager, product owners, Enterprise architect, team leads, Architect owners. Daily tactical huddle. DAD delivery team. Primary and secondary roles. Daily stand-up meeting, self-organising teams, empowerment.	
Frequent turnover within the project team	definitely
Self-contained DAD delivery team. Primary and secondary roles, including three leadership roles and separate HR manager. Self-organising teams, empowerment, task sharing, T-skilling.	

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
High level of technical complexity	definitely
Entire DAD delivery team involved in inception phase. Spikes. Provide story points and estimates. Sprint planning.	
Highly complex task being automated	definitely
Entire DAD delivery team involved in inception phase. Spikes. Provide story points and estimates. Sprint planning.	
Immature technology	definitely
DAD delivery teams are empowered to make tool decision. Temporary roles in DAD delivery teams. Coaching and training for upskilling on unfamiliar technology.	
Inadequate estimation of project budget	somewhat
Features are delivered through program.	
Inadequate estimation of project schedule	definitely
User stories and estimates. Priority setting. Spikes. User stories, story points. Re-estimating.	
Inadequate estimation of required resources	definitely
Self contained DAD delivery team with up to 13 individuals. Primary and secondary roles three leadership roles and separate HR manager. Empowered self-organising teams. Task sharing. T-skilling.	
Inadequately trained development team members	definitely
Project management including Product owner, Architecture owner. Self-organising teams, T-skilled, coaching by Architecture owner and Product owner, pair programming. Bring in outside expertise to develop and upskill team members.	
Ineffective communication	definitely
Project management: empowerment, self contained delivery team (PO, AO, TL, SE, QA, T. Writer etc), self-organising, pair programming.	
Ineffective project manager	definitely
Shared leadership roles (Product owner, Architecture owner, Technical lead) for DAD delivery teams. Project management tasks are shared by the entire DAD delivery team (individuals in primary roles).	
Inexperienced project manager	definitely
Shared leadership roles (Product owner, Architecture owner, Technical lead) for DAD delivery teams. Project management tasks are shared by the entire DAD delivery team (individuals in primary roles).	
Inexperienced team members	definitely
Self-contained DAD delivery team with up 13 individuals. Primary and secondary roles. Three leadership roles. Task sharing, pair programming.	
Lack of an effective project management methodology	definitely
Clearly defined program management roles and practices. Clearly defined project management practices and roles, including DAD method phases and activities.	
Lack of commitment to the project among development team members	definitely

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Project management including Team lead. Self-organising teams, inception phase engineers empowered to run spikes, provide story points and re-estimate each user story. Sprint planning and committment to sprint objectives.	
Lack of people skills in project leadership	definitely
Clearly defined leadership roles and practices with program and project management including based on self-organising DAD delivery teams.	
Large number of links to other systems required	not at all
Negative attitudes by development team	definitely
Self-organising teams. Inception phase engineers empowered to run spikes, provide story points, and re-estimate each user story. Sprint planning and committment to sprint objectives.	
new: Ineffective collaboration	definitely
All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles -every one are treated as equal, empowered, and make collective decisions. Three leadership roles with a DAD project team- team lead, AO and PO. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, pair programming. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles. Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, pair programming. Portfolio planning at enterprise level, product management, program planning at engineering level, PO, enterprise architects. Program and project levels- mandated retrospective practice for every program and iteration cycle in all DAD project phases. Program level- user story, a mandated practices which all project teams must adopted.	
new: Ineffective coordination	definitely
All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, iteration planning, iteration reviews. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, pair programming. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, pair programming. Iteration planning based on current team capacity Product manager, user experience engineer at program level and at project level PO, UAT, DOD. Program and project levels- work item list (captures the total amount of work done in program and project which includes bug fixes as well besides new feature implementation.	
new: Lack of trust	definitely

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles -every one are treated as equal, empowered, and make collective decisions. Any training need for the team is allowed through the secondary role- a coach, consultant or skilled individual member from another team can join for a period of time to upskill those who need upskilling. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles -every one are treated as equal, empowered, and make collective decisions. T-skilled means every team must learn on the fly all the broad skills required to deliver projects. All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles only-every one are treated as equal, empowered, and make collective decisions. However, AO, PO and Team lead can make decisions so that teams can make progress if stuck with a problem. DevOp practices at engineering level- must work closely, communicate and collaborate with operations team to learn about customer behaviour to feed in with design decisions. Governance practice by program management for all their project teams- DOD(definition of Done). Portfolio planning at enterprise level and program planning at engineering level- product managers, business case, enterprise architects, user experience engineers, PO. Program and project level planning- priority setting by product managers and product owners. Program and project levels- work item list (captures the total amount of work done in program and project which includes bug fixes as well besides new feature implementation. Project level- (DOD) definition of done practice defined by program management. Upfront architecture planning at enterprise level, program level and project levels (inception phase-run spikes, construction phase- look ahead planning in each iteration cycle). Team of enterprise architects and architect owners. Upfront architecture planning at enterprise level, program level and project levels. Team of enterprise architects and architect owners.	
One of the largest projects attempted by the organization	definitely
Program management, including Enterprise architect and product owners. Work item list and DAD delivery teams. Iteration show and tell.	
Poor project planning	definitely
Program Manager, Enterprise architects, Product owners. Work item list. Story boarding. Architecture planning. Product owners, Architect owners, Team lead, DAD delivery team. Inception phase, iteration planning, DOD, UAT. Re-estimation, story points.	
Project affects a large number of user departments or units	definitely
Product management, portfilio management, project management, IT operations including cloud deployment.	
Project involves the use of new technology	definitely
DAD delivery teams are empowered to make tool decision. Technical experts as temporary roles in DAD delivery teams. Coaching and training for upskilling on unfamiliar technology.	
Project involves use of technology that has not been used in prior projects	definitely
DAD delivery teams are empowered to make tool decisions. Temporary roles in DAD delivery teams. Coaching and training for upskilling on unfamiliar technology.	
Project milestones not clearly defined	definitely
Project management- each phase can be a milestone including creating a work item list, delivering sprint/iteration based on DOD.	
Project progress not monitored closely enough	definitely

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Program manager responsible to deliver a program. Program daily huddles. Iteration show and tell. Product owner, Architecture owner, team leader. User story points. Time-boxed construction phase has several iteration cycles, all time boxed. Daily stand-up meeting.	
Team members lack specialized skills required by the project	definitely
Product owner, Architecture owner. Self-organising teams, T-skilling, coaching by Architecture owner and Product owner, pair programming. Bring in outside expertise to develop and upskill team members.	
Team members not familiar with the task(s) being automated	definitely
Inception phase involving entire DAD delivery team. Spikes. Story points and estimates, sprint planning.	
Quadrant 4: Environment	
Change in organizational management during the project	definitely
IT governances drives DAD organisation structures. Product management, portfolio management, program management, and project management.	
Corporate politics with negative effect on project	definitely
Product management, portfolio management, program management and project management. Collective decision making at various levels with DAD framework.	
Dependency on outside suppliers	not at all
Many external suppliers involved in the development project	not at all
new: Country-specific regulations	definitely
DevOp practices- collaboration between Operations and SE (program and project)teams. Portfolio planning at enterprise level, product management, program planning at engineering level, PO, enterprise architects.	
new: Delays caused by global distance	definitely
DevOp practices- collaboration between Operations and SE (program and project)teams. Mandated DOD, TDD practices and including manual systems testing before deployment into production environment. Program management- program manager, enterprise architect, PO.	
new: Lack of architecture-organization alignment	definitely

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
DevOp practices at program and project levels. Performance monitoring practice of product environment for real time data for program portfolio planning and project planning. Enterprise architectures provide technology roadmap and guidance for project teams. Teams must run spikes. Enterprise architecture team (EAT) [VP architect + 8 enterprise architects (EA) from different geographic locations] plan and provide architecture strategy and guidance for the entire organization through weekly virtual EAT meetings. provide are Program architects owners form the enterprise architecture team" Feature funnel- portfolio planning at program level develop solution alternatives and write light weight business case by the enterprise architect (program architects). Inception phase (two week iteration) in all projects, run spikes on all user stories to minimize risks. Architecture Owner (AO) responsibility. Enterprise Architect will provide guidance and support. separate leadership roles in DAD development teams, there's a team lead role; there's a product owner who's representing product directions. And there's architecture owner (AO) who's responsible for ensuring we're looking at the architectural issues". Look ahead planning during iterations (construction phase). Program and project level upfront planning. Program portfolio planning and project planning (inception phase). Enterprise architects and Architect owners responsibilities. Program manager and enterprise architects, coordinate and have face to face meeting with all project happening across the various geographic location under their programs. The enterprise architecture team's (EAT) goal is to collaborate across all those boundaries with other EAs in the EAT". One of the functions of the enterprise architecture team to be able to share that information in both direction, locally make sure everyone is aware off it, let people locally know what is happening with other teams and other architects.	
new: Lack of face-to-face interaction inhibits knowledge sharing	definitely
All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, iteration planning, iteration reviews. DevOp practices- collaboration between Operations and SE (program and project)teams. Project level (DAD project teams)- inception phase-spikes, story-points and re-estimate, construction phase- look ahead planning, iteration planning- re-estimate	
new: Lack of process alignment	definitely
All DAD project teams are feature team, self-contained and self-organizing teams, T-skilled and all team members are in primary roles (software engineers). Three leadership roles with a DAD project team- team lead, AO and PO. Co-location, shared workspace, daily stand-up meetings, iteration planning, iteration reviews. Flexibility and empowerment for project teams to also adopt their team specific practices. DAD project teams- a coach or consultant in secondary role Product management and program management- mandated practices for projects. Project level- there phases inception, construction and hardening phases. All based on two week iteration cycle (construction- will have several iteration cycles)- flexibility with duration for each key activity within a phase.	
new: Lack of tool/infrastructure alignment	definitely
Booking shared resources. Mandated DevOp practice, collaboration between operations and engineering teams. Mandated practice- DevOp practice, collaboration between operations and engineering teams. Mandated practices for projects by program management for TDD and automation. Program management and project Management- leadership roles guide, support and provide knowledge and understanding. TDD and automation T-skilled engineers innovate to solve problems. T-skilled engineers learn on the fly.	
new: Unstable country/regional political/economic environment	not at all
Organization undergoing restructuring during the project	not at all
Resources shifted from the project due to changes in organizational priorities	definitely

Continued on next page

Table 3 (continued): DAD practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Program level defines projects and resources. DAD delivery teams remain fixed based on small, medium and large teams (each has a defined number of individuals).	
Unstable organizational environment	definitely
IT governances drives DAD organisation structures. Product management, portfolio management, program management, and project management.	

3.3 SAFe Practice Mapping to GSD Risk Catalog risks

Table 4: SAFe practices mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Practices	*Level*
Quadrant 1: Customer Mandate	
Conflict between users	somewhat
Help to derive collaboration among the key stakeholders in the business. Use capabilities as end-to-end solution services that support the achievement of user goals. Complete user stories from Team Backlog and assure that each story meets its (local) Definition of Done. Establish team backlog by adding user stories, enabler stories, and improvement stories. Capabilities are the end-to-end solution services that support the achievement of user goals. Use models to predict performance (response time, reliability) or physical properties (heat, radiation, strength), or explore designs for user experience or response to an external stimulus.	
Lack of cooperation from users	definitely
Encourage and facilitate incremental development and fast customer feedback. Facilitate continuous improvement by quantitative metrics, customer feedback, and the Inspect and Adapt retrospective cycle. Engage stakeholders and provide feedback–time-boxed to an hour or less. Use capabilities as end-to-end solution services that support the achievement of user goals. Fast and intimate feedback from customers. Release more frequently to gain the meaningful feedback about efficacy, deployability, and usability of the product (Each release helps assess value of product in the development environment). Identify user-business value. User Experience Designer (UX). Work with stakeholders to understand the specific business targets behind the user-system interaction. Provide Agile teams with the next increment of UI design, User experience guidelines, and design elements in a just-in-time (but timely enough) fashion. Continuously validate user experience via user experience testing. Work with system and solution architect/engineering and teams to build and maintain the technical foundation for real-time user experience validation, feedback, tracking statistics, etc. Share user experience guidelines across the program; educate developers on the best practices of maintaining good UI design. Assist test engineers and the System team in user experience testing and test automation. Lead UI design and User experience/UI community of practice workshop. Develop "user stories that are value centric and focus on the user not the system". Demonstrate enabler stories by showing the artifacts produced. Use the Conversation" between the team, customers, user, or other stakeholders to determine more detailed behaviors required to implement intent". Provides "feedback from the stakeholders about the efficacy and usability of the system under development". Ensure that integration between teams on the same ART occurs on a regular basis and no less than every iteration. Capabilities are the end-to-end solution services that support the achievement of user goals. Perform continuous exploration that includes learning milestones, customer feedback loops, and set-based design, which informs and streamlines the learning process by validating good options and eliminating less viable ones.	
Lack of top management support for the project	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Allocate budget for development of value stream. Define Strategic Themes. Allocate budget authority to the decision makers. Value streams budgets. Provides the spending and personnel allocations necessary to accomplish the portfolio vision. Fund Value Streams, Not Projects. Portfolio backlog.	
Lack of user participation	definitely
Facilitate continuous improvement by quantitative metrics, customer feedback, and the Inspect and Adapt retrospective cycle. Develop a feature team that is organized around user-centered functionality. Each team is capable of delivering end-to-end value. Feature teams operate primarily with user stories, refactors, and spikes. Fast and intimate feedback from customers. Update enabler definitions and models to develop use cases that illustrate how the features and capabilities work together to deliver the end user value. Identify user-business value. Continuously communicate emerging requirements and opportunities back into the program vision through product owner. Create platforms and environments for solution demonstration, QA, user testing, etc. User Experience Designer (UX). Work with stakeholders to understand the specific business targets behind the user-system interaction. Continuously validate user experience via user experience testing. Share user experience guidelines across the program; educate developers on the best practices of maintaining good UI design. Lead UI design and User experience/UI community of practice workshop. Use the Conversation" between the team, customers, user, or other stakeholders to determine more detailed behaviors required to implement intent". Final performance test includes Nonfunctional Requirements (NFR) testing, standards and security validations, user acceptance testing, final documentation, and any other readiness activities that are not feasible or economical to perform at every iteration. Show progress that the solution has made during the past program increment to value stream stakeholders, Customers (or their internal proxies), and senior management. Establish team backlog by adding user stories, enabler stories, and improvement stories. Balance the backlog of internally facing work with new user stories that deliver value. Solution Demo includes the following stakeholders: Solution Management, Value Stream Engineer, Architects/Engineering, Customers, ART representatives, Program Portfolio Management representatives, Value Stream Level stakeholders, executive sponsors, senior management, and DevOps.	
Lack or loss of organizational commitment to the project	definitely
Allocate budget for development of value stream. Define Strategic Themes. Allocate budget authority to the decision makers. Value streams budgets. Provides the spending and personnel allocations necessary to accomplish the portfolio vision. Fund Value Streams, Not Projects. Portfolio backlog. Ensure that the organization's release governance is understood.	
Users not committed to the project	definitely
Encourage and facilitate incremental development and fast customer feedback. Facilitate continuous improvement by quantitative metrics, customer feedback, and the Inspect and Adapt retrospective cycle. Engage stakeholders and provide feedback–time-boxed to an hour or less. Use capabilities as end-to-end solution services that support the achievement of user goals. Fast and intimate feedback from customers. Release more frequently to gain the meaningful feedback about efficacy, deployability, and usability of the product (Each release helps assess value of product in the development environment). Identify user-business value. Develop "user stories that are value centric and focus on the user not the system". Demonstrate enabler stories by showing the artifacts produced. Use the Conversation" between the team, customers, user, or other stakeholders to determine more detailed behaviors required to implement intent". Provides "feedback from the stakeholders about the efficacy and usability of the system under development". Ensure that integration between teams on the same ART occurs on a regular basis and no less than every iteration. Capabilities are the end-to-end solution services that support the achievement of user goals. Perform continuous exploration that includes learning milestones, customer feedback loops, and set-based design, which informs and streamlines the learning process by validating good options and eliminating less viable ones.	

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Users resistant to change	somewhat
Provide the opportunity for 'just the right amount' of Architecture and User Experience guidance. Identify user-business value. Continuously validate user experience via user experience testing. Develop user stories to "express needed functionality" (Replace the traditional requirement specification). Develop "user stories that are value centric and focus on the user not the system". Consider the user might be the end user or a device or a another system. Demonstrate enabler stories by showing the artifacts produced. Use the Card" to capture the statement of intent of the user story (Index card, sticky note or tool)". Use the Conversation" between the team, customers, user, or other stakeholders to determine more detailed behaviors required to implement intent". Compare value of three things: defects; refactors, redesigns, technology upgrades; new user stories.	
Users with negative attitudes toward the project	definitely
Encourage and facilitate incremental development and fast customer feedback. Facilitate continuous improvement by quantitative metrics, customer feedback, and the Inspect and Adapt retrospective cycle. Engage stakeholders and provide feedback–time-boxed to an hour or less. Use capabilities as end-to-end solution services that support the achievement of user goals. Fast and intimate feedback from customers. Release more frequently to gain the meaningful feedback about efficacy, deployability, and usability of the product (Each release helps assess value of product in the development environment). Identify user-business value. Develop "user stories that are value centric and focus on the user not the system". Demonstrate enabler stories by showing the artifacts produced. Use the Conversation" between the team, customers, user, or other stakeholders to determine more detailed behaviors required to implement intent". Provides "feedback from the stakeholders about the efficacy and usability of the system under development". Ensure that integration between teams on the same ART occurs on a regular basis and no less than every iteration. Capabilities are the end-to-end solution services that support the achievement of user goals. Perform continuous exploration that includes learning milestones, customer feedback loops, and set-based design, which informs and streamlines the learning process by validating good options and eliminating less viable ones.	
Quadrant 2: Scope and Requirements	
Conflicting system requirements	definitely
Apply enabler for exploration that provides a way for development teams to flesh out the details of requirements and design. Continuously communicate emerging requirements and opportunities back into the program vision through product owner. Work with stakeholders to understand the specific business targets behind the user-system interaction. Develop user stories to "express needed functionality" (Replace the traditional requirement specification). Solution intent records and communicates requirements, design, and system architecture decisions. Adopt requirements and modeling tools such as Model-Based System Engineering (MBSE). Replace textual requirements with automated acceptance tests for features and stories where possible. Seek executable specifications (ATDD- Acceptance Test-Driven Development), where requirements are specified in a form that can be executed for testing. Make requirements and tests one and the same where possible, and automate to the extent possible. Adopt variable intent that represents the elements for which system builders are free to explore the economic trade-offs of requirements and design alternatives that could meet the need.	
Continually changing project scope/objectives	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Perform system demo as near as possible to the end of the iteration. Integrate every other iteration. Actively participate in ongoing agreements to maintain business and development alignment as priorities and scope are inevitably changed. Use "iteration retrospective to drive program level changes to process either immediately or in the Inspect and Adapt workshop". Perform Iteration Planning before each iteration. Adopt quality practices that allow engineers to confidently and frequently make model changes and contribute to the system intent. Adopt Test-First practices to help teams to build quality into their products early and facilitate the continuous and small changes.	
Continually changing system requirements	definitely
Perform system demo as near as possible to the end of the iteration. Integrate every other iteration. Continuously communicate emerging requirements and opportunities back into the program vision through product owner. Actively participate in ongoing agreements to maintain business and development alignment as priorities and scope are inevitably changed. Work with stakeholders to understand the specific business targets behind the user-system interaction. Use "iteration retrospective to drive program level changes to process either immediately or in the Inspect and Adapt workshop". Perform Iteration Planning before each iteration. Adopt quality practices that allow engineers to confidently and frequently make model changes and contribute to the system intent. Adopt Test-First practices to help teams to build quality into their products early and facilitate the continuous and small changes.	
Difficulty in defining the inputs and outputs of the system	definitely
Define PI planning's primary outputs. Demonstrate each new feature in an end-to-end use case. Develop a feature team that is organized around user-centered functionality. Each team is capable of delivering end-to-end value. Feature teams operate primarily with user stories, refactors, and spikes. Use capabilities as end-to-end solution services that support the achievement of user goals. Define inputs to the solution vision. Epic Owner works with development team to size the epic and provide input for economic prioritization based on WSJF. Perform end-to-end and solution performance testing. Solution Demo demonstrate each objective and capability in an end-to-end use case. Capabilities are the end-to-end solution services that support the achievement of user goals. Implement capabilities via vertical, end-to-end slices of value, which enable incremental solution development.	
Ill-defined project goals	definitely
Implement value stream coordination to ensure that the enterprise moves forward with each value stream in lockstep with the enterprise objectives. Align development to business via business context, vision, and Team and Program PI Objectives. Create a set of 'SMART' team PI objectives for each individual team with business value assigned. Use capabilities as end-to-end solution services that support the achievement of user goals. SM facilitates the team's progress toward the goal. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Develop Program Increment (PI) objectives. Build Team PI Objectives. Build PI Objectives during PI Planning -ensuring the following is in place;. Differentiate between Features and Objectives. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives". Write SMART Objectives (Specific, Measureable, Achievable, Realistic, Time-bound). (S) Provide a concise and simple description of the intended outcome (usually starting with an action verb). (M)easure what a team needs to do to achieve the team objectives (can be descriptive; yes or no; or quantitative; or within a range). (A) objectives should be achievable i.e; within the team's control and influence. (T)ime "period for achievement must be within the PI, and all objectives must be scoped appropriately". Understand and agree on one or more iteration goals that are based on the team and PI planning objectives.	
Incorrect system requirements	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Epics and lightweight business cases. Ensure that Epics and Enablers are reasoned and analyzed prior to reaching a Program Increment boundary, are prioritized appropriately, and have established acceptance criteria to guide a high-fidelity implementation. Define Roadmap. Produce system demo for program and value stream stakeholders. Produce the PI system demo for program stakeholders. Participate in the preparation of the PI system demo to make sure that they will be able to show the most critical aspects of the solution to the stakeholders. Perform system and solution demos. Work with system and solution architect/engineering and teams to build and maintain the technical foundation for real-time user experience validation, feedback, tracking statistics, etc. Work with customers, stakeholders, and suppliers to establish high-level solution intent; help establish the solution intent information models and documentation requirements. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Prepare a "system demo and provide time for final testing of the Solution". Team plans and commits to a set of PI objectives together. Develop Program Increment (PI) objectives.	
System requirements not adequately identified	definitely
Epics and lightweight business cases. Ensure that Epics and Enablers are reasoned and analyzed prior to reaching a Program Increment boundary, are prioritized appropriately, and have established acceptance criteria to guide a high-fidelity implementation. Define Roadmap. Apply enabler for exploration that provides a way for development teams to flesh out the details of requirements and design. Integrate to illustrate a particular feature, capability, or nonfunctional requirement. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Invest in Good stories. Team plans and commits to a set of PI objectives together. Develop Program Increment (PI) objectives.	
Unclear system requirements	definitely
Epics and lightweight business cases. Ensure that Epics and Enablers are reasoned and analyzed prior to reaching a Program Increment boundary, are prioritized appropriately, and have established acceptance criteria to guide a high-fidelity implementation. Define Roadmap. System and Solution Architect /Engineer responsibilities. System and Solution Architect /Engineer participates in planning, definition, and high-level design of the solution and explore solution alternatives. System and Solution Architect /Engineer defines subsystems and their interfaces; System and Solution Architect /Engineer allocates responsibilities to subsystems; System and Solution Architect /Engineer understands solution deployment, System and Solution Architect /Engineer communicates requirements for interactions with solution context. System and Solution Architect /Engineer role involves four practices:. System and Solution Architect /Engineer defines, explores, and supports the implementation of value stream and program enablers to evolve solution intent; work directly with Agile Teams to implement, explore, or support them. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Consider who is using the system and what specifically they doing with it and why they are doing it. Place PI system demo, inspect and adapt workshop, and PI planning in a dedicated IP iteration. Team plans and commits to a set of PI objectives together. Develop Program Increment (PI) objectives.	
Undefined project success criteria	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Define Success criteria to validate the implementation. Impacts the identification, success criteria, and prioritization of epics in the funnel and backlog states. Success criteria provide a mechanism to understand progress towards the intent. Success criteria provide learning milestones that allow the portfolio to understand the solution involved, validate business and technical hypotheses, and pivot towards a better solution. Represents the current estimated story points, rolled up from the epic's child features/stories and the initial epic estimate. Use capabilities as end-to-end solution services that support the achievement of user goals. Define epic success criteria. SM facilitates the team's progress toward the goal. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives". Understand and agree on one or more iteration goals that are based on the team and PI planning objectives.	
Users lack understanding of system capabilities and limitations	definitely
Perform system demos. Integrate to illustrate a particular feature, capability, or nonfunctional requirement. Demonstrate each new feature in an end-to-end use case. Perform a final system demo of all features developed in the PI during the 'Inspect and Adapt workshop' at the PI boundary. Perform system demo after each system increment. Direct and guide through value stream capabilities backlog to the vision. Participate and provide feedback from the solution demos relevant to the capabilities and subsystem being built by the ART. Create platforms and environments for solution demonstration, QA, user testing, etc. Participate in PI planning and the pre- and post- PI planning meetings at the value stream level, and in backlog refinement to define integration and test capabilities and features. Solution Demo includes the following stakeholders: Solution Management, Value Stream Engineer, Architects/Engineering, Customers, ART representatives, Program Portfolio Management representatives, Value Stream Level stakeholders, executive sponsors, senior management, and DevOps.	
Quadrant 3: Execution	
Development team unfamiliar with selected development tools	somewhat
Understand requirements for working on "technical infrastructure, tooling, and other systemic impediments". It may be more efficient to perform an upgrade or migration at a time when there isn't a critical demo just a few days away. Perform Daily Stand-up to understand team's status, "escalate problems, and get help from other team members". Recognize "stretch objectives are not the way for stakeholders to load the teams with more than they can do".	
Frequent conflicts among development team members	somewhat
Vote of confidence/commitment from the entire program to these objectives. Develop PI commitment. Establish an agreement to determine how the work is performed for each activity type. Actively circulate during planning, communicate business priorities to the teams, and maintain agreement and alignment among the stakeholders as to the key objectives of the train. Actively participate in ongoing agreements to maintain business and development alignment as priorities and scope are inevitably changed. Team plans and commits to a set of PI objectives together. Commit to PI Objectives by agreeing to do everything in the team's power to meet the committed objectives. Continuous collaboration integrates the entire system-of-systems value stream to demonstrate progress toward the top-level context's Milestone and Release commitments.	
Frequent turnover within the project team	not at all
High level of technical complexity	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Break down the architectural enabler into small enabler stories that can fit in iterations. Perform initial exploration of epics and rank them roughly by using Weighted Shortest Job First (WSJF) to determine which epics should move to the next step for deeper exploration. Apply Weighted Shortest Job First prioritization method for job sequencing. Define Weighted Shortest Job First (WSJF). Discuss each story in terms of "relative difficulty, size, complexity and technical challenges, and establish acceptance criteria". Break each story into tasks then estimate and identify dependencies of a specific task. Break larger initiatives into stories to ease estimation of larger work items. Use either size/value or Weighted Shortest Job First (WSJF) to prioritize the stories issued by product owner.	
Highly complex task being automated	definitely
Perform initial exploration of epics and rank them roughly by using Weighted Shortest Job First (WSJF) to determine which epics should move to the next step for deeper exploration. Apply Weighted Shortest Job First prioritization method for job sequencing. Define Weighted Shortest Job First (WSJF). Extend test scenarios to larger data sets. Perform manual testing and run automated tests for new features and stories. PI Features are broken into stories and placed on team backlog. Use either size/value or Weighted Shortest Job First (WSJF) to prioritize the stories issued by product owner.	
Immature technology	definitely
Work with Agile Teams that perform research spikes, create proof of concepts, mock-ups, etc. Support technology/engineering aspects of program and value stream kanbans. Local stories representing new functionality, refactors, defects, research spikes, and other technical debt are identified, written as enabler stories, estimated, and sequenced. Compare value of three things: defects; refactors, redesigns, technology upgrades; new user stories. Include enablers that could constitute infrastructure work, refactoring, research spikes, architectural improvement, and defects.	
Inadequate estimation of project budget	definitely
Develop Incremental implementation by keeping the epics in the portfolio backlog until there is implementation capacity available. Avoid overhead and enables the train to make fast and local decisions within the constraints of the allocated budget. Provide WIP limits to ensure that the teams responsible for analysis undertake into responsibly and do not create expectation for implementation or time frames that far exceed capacity and reality. Use capacity allocation to estimate portfolio epic based on the given knowledge of program velocities. Match demand to capacity and eliminate excess WIP. Apply capacity allocation for enabler to work as a whole or to differentiate between various types of enablers. Apply capacity allocation to make a decision about how much of the total effort can be applied for each type of activity for an upcoming PI. The decision can be revisited as part of backlog refinement in preparation of each PI planning. Understand and operate within the ART Budget. System and Solution Architect/Engineer works with Product and Solution Management to determine capacity allocation for enablement work. Provides necessary clarifications to assist team with their story estimation and story sequencing for the upcoming program increment. System and Solution Architect /Engineer works with product and solution management to determine capacity allocation for enablement work. Measure the team's capacity based on previous sprint to establish the velocity for the upcoming iteration. Value Stream and Program PI events synchronize customer feedback, resource and budget adjustments [Use synchronized events to facilitate cross-functional trade-offs]. Estimate initial capacity to establish a common starting point for team estimation. Apply Capacity allocation technique to make a policy decision as to how much total effort apply to each of the three activities listed. Establish velocity by quantifying team capacity. Identify stretch objectives to provide the flexible capacity and scope management options needed to increase reliability and quality of PI execution.	
Inadequate estimation of project schedule	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Agile Estimating and Planning. Adopt Agile estimating and planning by using the currency of story points. Epic Progress Measure. Identify, elaborate, estimate, and analyze all epics that reached in mature state to achieve a go recommendation from PPM. Use capacity allocation to estimate portfolio epic based on the given knowledge of program velocities. Apply normalized estimation technique during the backlog refinement or equivalent approach used by the Agile Teams to estimate stories. Estimate Longer-Term Initiatives. Use Agile story point physics, based on normalized estimation, and estimate larger initiatives at the Epic level. Move features that align with the vision and support strategic themes to feature refinement for further exploration. General sizing of features also occurs at this step, as features are estimated in normalized story points. (The purpose of such estimation is to support economic estimating and forecasting of value delivery over a longer period of time based on scope.). Work with the team to establish technical feasibility and scope estimates. Assist with economic decision-making by facilitating feature and capability estimation by teams and roll-up to the value stream level and Portfolio level. Supports estimation. Break each story into tasks then estimate and identify dependencies of a specific task. Planning at regular Program Increment (PI) intervals limits variances to a single PI time-box, thereby increasing Agile Release Train and Value Stream predictability [Use a regular cadence to limit the accumulation of variance]. Estimate Work. Estimate initial capacity to establish a common starting point for team estimation. Break larger initiatives into stories to ease estimation of larger work items. Pull jobs into implementation based on WSJF, where estimate of job size is typically used as a proxy for duration.	
Inadequate estimation of required resources	definitely
Develop Incremental implementation by keeping the epics in the portfolio backlog until there is implementation capacity available. Provide WIP limits to ensure that the teams responsible for analysis undertake into responsibly and do not create expectation for implementation or time frames that far exceed capacity and reality. Use capacity allocation to estimate portfolio epic based on the given knowledge of program velocities. Match demand to capacity and eliminate excess WIP. Apply capacity allocation for enabler to work as a whole or to differentiate between various types of enablers. Apply capacity allocation to make a decision about how much of the total effort can be applied for each type of activity for an upcoming PI. The decision can be revisited as part of backlog refinement in preparation of each PI planning. Assist with economic decision-making by facilitating feature and capability estimation by teams and roll-up to the value stream level and Portfolio level. Actively circulate during planning, communicate business priorities to the teams, and maintain agreement and alignment among the stakeholders as to the key objectives of the train. System and Solution Architect/Engineer works with Product and Solution Management to determine capacity allocation for enablement work. System and Solution Architect /Engineer works with product and solution management to determine capacity allocation for enablement work. Measure the team's capacity based on previous sprint to establish the velocity for the upcoming iteration. Team approve any scope, timing, or resource adjustments necessary to help ensure the release. Value Stream and Program PI events synchronize customer feedback, resource and budget adjustments [Use synchronized events to facilitate cross-functional trade-offs]. Estimate initial capacity to establish a common starting point for team estimation. Apply Capacity allocation technique to make a policy decision as to how much total effort apply to each of the three activities listed. Establish velocity by quantifying team capacity. Identify stretch objectives to provide the flexible capacity and scope management options needed to increase reliability and quality of PI execution.	
Inadequately trained development team members	somewhat
Conduct specialized training to keep up with advancements in their respective fields.	
Ineffective communication	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Facilitate continuous improvement by quantitative metrics, customer feedback, and the Inspect and Adapt retrospective cycle. Establish high-bandwidth communication across all team members and stakeholders. Perform system demo as near as possible to the end of the iteration. Balance integration effort and feedback. Integrate to illustrate a particular feature, capability, or nonfunctional requirement. Integrate every other iteration. Allow questions and comments by opening a forum. Summarize progress, feedback, and action items. Perform Retrospective and problem-solving workshop. Run a brief retrospective to identify whatever issues team would like to address. Perform a final system demo of all features developed in the PI during the 'Inspect and Adapt workshop' at the PI boundary. Perform system demo after each system increment. Attend System Demos and Solution Demos. Attend Solution and possibly System Demo; help evaluate the solution increment. Coordinate with marketing and with product and solution management on internal and external communications. Perform Daily Stand-up in front of the BVIR that highlights the stories. "SAFe teams plan together, integrate and demo together, and learn together". Perform Iteration Retrospective as the check step for the overall iteration. Use "iteration retrospective to drive program level changes to process either immediately or in the Inspect and Adapt workshop". Use Conversation" throughout the story life cycle to include backlog refinement planning implementation and demonstration". Place PI system demo, inspect and adapt workshop, and PI planning in a dedicated IP iteration. Demo these integrations at the system demo at the end of every iteration. Perform daily Stand-up meetings for team coordination. Demonstrate value and process improvement. Provide governance for any upcoming Releases and also provide regular communication to management. Demonstrate completed story and summarize team's increment. Perform Iteration Retrospective to identify way to improve. Perform daily stand-ups. Perform Team Demo at the end of each iteration. Perform retrospective at the end of each iteration.	
Ineffective project manager	not at all
Inexperienced project manager	not at all
Inexperienced team members	somewhat
Involve a subject matter expert in basic exploration and sizing. Interact with analyst and subject matter experts during specification workshops. Work with stakeholders and subject matter experts to define the epic and its potential benefits. Interact with analysts and subject matter experts during specification workshops. Estimate stories based on Volume - (How much is there?), Complexity - (How hard is it?), Knowledge - (What's known?), and Uncertainty - (What's not known?). Adopt Communities of Practice to foster and support continuous learning. Team approve any scope, timing, or resource adjustments necessary to help ensure the release.	
Lack of an effective project management methodology	definitely
Coach leaders, teams, and Scrum masters in lean-Agile practices and mindsets. Use "Agile Project Management Tools to capture stories and status, defects, test cases, estimates, actuals, assignments, burn-down chart".	
Lack of commitment to the project among development team members	definitely
Vote of confidence/commitment from the entire program to these objectives. Develop PI commitment. Limit the commitments to longer-term work, because some other item may come along that's more important than a prior commitment. Team plans and commits to a set of PI objectives together. Commit to PI Objectives by agreeing to do everything in the team's power to meet the committed objectives. Continuous collaboration integrates the entire system-of-systems value stream to demonstrate progress toward the top-level context's Milestone and Release commitments.	
Lack of people skills in project leadership	somewhat

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Define Scrum Master role. Exhibits Lean-Agile leadership. Protects and communicates. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives".	
Large number of links to other systems required	definitely
Perform initial exploration of epics and rank them roughly by using Weighted Shortest Job First (WSJF) to determine which epics should move to the next step for deeper exploration. Apply Weighted Shortest Job First prioritization method for job sequencing. Define Weighted Shortest Job First (WSJF). RTE helps to manage risks and dependencies. Trace significant external commitments and dependencies. Coordinate content dependencies with other product owners by attending weekly PO sync meetings. Use either size/value or Weighted Shortest Job First (WSJF) to prioritize the stories issued by product owner. Manage dependencies and resolve impediments by continuously and actively engaging with other teams.	
Negative attitudes by development team	somewhat
Perform Iteration Retrospective as the check step for the overall iteration. Use "iteration retrospective to drive program level changes to process either immediately or in the Inspect and Adapt workshop". Perform Iteration Retrospective to identify way to improve. Perform retrospective at the end of each iteration.	
new: Ineffective collaboration	definitely
Manage dependencies by applying extensive degree of cooperation; a common value stream backlog; implementation of new, crosscutting capabilities; additional system integration; additional roles and responsibilities; special considerations for pre-, post-, and PI planning activities; different degree and types of DevOps support. Product manager create features in collaboration with product owner and other key stakeholders. Features are also created as a result of decomposition of epics. Encourage the collaboration between teams and System and Solution Architects, Engineering, and User Experience designers. Value Stream Engineer encourages the collaboration between teams and System and Solution Architects, Engineering, and User Experience designers. Facilitate the technical aspects of collaboration with third parties, such as data or service providers, hosting facilities, etc. Use webcams, instant messaging, and other collaboration tools if teams are distributed. "SAFe teams plan together, integrate and demo together, and learn together". PI planning, iteration planning, backlog refinement, inspect and adapt, architecture discussion, etc., all benefits from frequent meetings [Schedule frequent meetings using a predictable cadence]. Sub practice: Encourage intense collaboration within and between teams, (recommendations):. Perform constant communication and collaboration Identify feature that requires collaboration. Improve continuous collaboration by using regular feedback loops. Bring together stakeholders from all parts of the value stream. Collaborate and integrate with the others to provide the operational value stream with a seamless, end-to-end solution. Continuous Collaboration Ensures Deployability. Continuous collaboration ensures deployability through continuous feedbacks. Continuous collaboration integrates the entire system-of-systems value stream to demonstrate progress toward the top-level context's Milestone and Release commitments. Continuous collaboration helps ensure that the solution can be deployed in the ultimate Customer's context. Effective Customer/system builder collaboration helps ensure that the system meets the Customers' needs. Facilitates cross-discipline collaboration by allowing traceability from a model in one discipline to a model in another.	
new: Ineffective coordination	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Manage dependencies by applying extensive degree of cooperation; a common value stream backlog; implementation of new, crosscutting capabilities; additional system integration; additional roles and responsibilities; special considerations for pre-, post-, and PI planning activities; different degree and types of DevOps support. Implement value stream coordination to ensure that the enterprise moves forward with each value stream in lockstep with the enterprise objectives. Align development to business via business context, vision, and Team and Program PI Objectives. Identify dependencies and forecast cross-team and cross-ART coordination. Identify dependencies and forecast cross-team and cross-ART coordination. Release on the program increment cadence. Develop on Cadence, Release Any Time. Support the goals of synchronization, alignment, management of variability, and predictability of development velocity. Move features that align with the vision and support strategic themes to feature refinement for further exploration. General sizing of features also occurs at this step, as features are estimated in normalized story points. (The purpose of such estimation is to support economic estimating and forecasting of value delivery over a longer period of time based on scope.). Coordinate content dependencies with other product owners by attending weekly PO sync meetings. Treated like an Agile Release Train and works in the same cadence as the other ARTs. Perform daily Stand-up meetings for team coordination. Develop Cadence Principles. Apply Cadence Principles. Planning at regular Program Increment (PI) intervals limits variances to a single PI time-box, thereby increasing Agile Release Train and Value Stream predictability [Use a regular cadence to limit the accumulation of variance]. Short iterations help to control the number of stories in the iteration batch [Use a regular cadence to enable small batch sizes]. PI planning, iteration planning, backlog refinement, inspect and adapt, architecture discussion, etc., all benefits from frequent meetings [Schedule frequent meetings using a predictable cadence]. Apply Synchronization Principles. Individual Agile Teams are aligned to common iteration lengths [Exploit economic of scale by synchronizing work from multiple projects]. Teams plan with stretch objectives; these are sacrificed as necessary when plan meet reality [Capacity margin enables synchronization of deliverables]. Teams are aligned to common time-boxes and similar batch sizes [To reduce queues, synchronize the batch size and timing of adjacent processes]. Teams integrate and evaluate (at least) on iteration boundaries; program and value streams integrate and evaluate on PI boundaries. [Apply nested cadence harmonic multiples to synchronize work]. Common Cadence. Use stretch objectives provide capacity margins required to synchronize to a delivery cadence. Identify dependencies and foster cross-ART coordination. Track changes as both the system and deployment environment have to evolve to a common state. Perform continuous Integration and testing while Solution Demonstration occurs on a fixed PI cadence. Coordinate among Solution Management and Solution Architect/Engineering to delegate some solution intent requirements directly to the ARTs that build the capabilities and subsystems of the solution.	
new: Lack of trust	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Manage dependencies by applying extensive degree of cooperation; a common value stream backlog; implementation of new, crosscutting capabilities; additional system integration; additional roles and responsibilities; special considerations for pre-, post-, and PI planning activities; different degree and types of DevOps support. Adopt SAFe's set of seven transformational patterns that can be used to move the organization to Lean-Agile Program Portfolio Management ([Decentralized decision-making], [Demand management; continuous value flow], [Lightweight; epic-only business cases], [Decentralized, rolling-wave planning], [Agile estimating and planning], [Lean-Agile budgeting and self-managing Agile Release Trains], [Objective, fact-based measure and milestones]). Self-managing Agile Release Trains. Adopt value-stream based, self-managing Agile Release Trains to provide a continuous of flow to its stakeholders. Identify current risks and impediments. Run a brief retrospective to identify whatever issues team would like to address. Identify risk reduction-opportunity enablement value. Discuss with team if team members find themselves overcommitted. Agree on the final list of stories that will be achieved, and revisit and restate the iteration goals (Both product owner and team members). Visualize the status of the stories, defects, and other activities that the team is working on during the iteration. "Ensure that teams don't focus solely on local optimization". "SAFe teams plan together, integrate and demo together, and learn together". "Team show a tested increment of value to product owner". Team receive feedback from product owner on what they produced. Team evaluates its process and any improvement stories it had from previous iteration. Team identify problems and root causes as well as bright spots. "Team come up with improvement stories that enter the team for the next iteration". Each "team member takes responsibility for a specific task or tasks". Perform daily Stand-up meetings for team coordination. Consider upgrading and enhancing "intra- and inter-team communications systems". Perform Daily Stand-up to understand team's status, "escalate problems, and get help from other team members". Team approve any scope, timing, or resource adjustments necessary to help ensure the release. Ensure that integration between teams on the same ART occurs on a regular basis and no less than every iteration. Demonstrate completed story and summarize team's increment. Teams plan with stretch objectives; these are sacrificed as necessary when plan meet reality [Capacity margin enables synchronization of deliverables]. Develop iteration goals to align team members to a common purpose. Develop iteration goals to align program teams to common PI Objectives and Manage dependencies. Sub practice: Encourage intense collaboration within and between teams, (recommendations):. Perform daily stand-ups. Perform retrospective at the end of each iteration. Team plans and commits to a set of PI objectives together. Team learn together and share best practices through inter-team communication supported by Communities of Practice (CoP). Build Team PI Objectives. Manage dependencies and resolve impediments by continuously and actively engaging with other teams. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives". Team agree to do their best to deliver the both non-stretch and stretch objectives. Recognize "stretch objectives are not the way for stakeholders to load the teams with more than they can do". Understand and agree on one or more iteration goals that are based on the team and PI planning objectives. Empower people particularly Product and Solution Management with the relevant context, knowledge, and authority to make content decision at each level. Encourages general knowledge discovery by making information, and related cross-discipline information, more accessible to teams. Establish the solution intent's organizational structure and define where various types of information are managed to support analysis and compliance needs.	
One of the largest projects attempted by the organization	definitely
Perform initial exploration of epics and rank them roughly by using Weighted Shortest Job First (WSJF) to determine which epics should move to the next step for deeper exploration. Apply Weighted Shortest Job First prioritization method for job sequencing. Define Weighted Shortest Job First (WSJF). Short iterations help to control the number of stories in the iteration batch [Use a regular cadence to enable small batch sizes]. Use either size/value or Weighted Shortest Job First (WSJF) to prioritize the stories issued by product owner.	

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Poor project planning	definitely
Implement value stream coordination to ensure that the enterprise moves forward with each value stream in lockstep with the enterprise objectives. Align development to business via business context, vision, and Team and Program PI Objectives. Create a set of 'SMART' team PI objectives for each individual team with business value assigned. Apply normalized estimation technique during the backlog refinement or equivalent approach used by the Agile Teams to estimate stories. Use Agile story point physics, based on normalized estimation, and estimate larger initiatives at the Epic level. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Planning at regular Program Increment (PI) intervals limits variances to a single PI time-box, thereby increasing Agile Release Train and Value Stream predictability [Use a regular cadence to limit the accumulation of variance]. Estimate initial capacity to establish a common starting point for team estimation. Break larger initiatives into stories to ease estimation of larger work items. Develop Program Increment (PI) objectives. Build Team PI Objectives. Build PI Objectives during PI Planning -ensuring the following is in place;. Differentiate between Features and Objectives. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives". Write SMART Objectives (Specific, Measureable, Achievable, Realistic, Time-bound). (S) Provide a concise and simple description of the intended outcome (usually starting with an action verb). (M)easure what a team needs to do to achieve the team objectives (can be descriptive; yes or no; or quantitative; or within a range). (A) objectives should be achievable i.e; within the team's control and influence. (T)ime "period for achievement must be within the PI, and all objectives must be scoped appropriately".	
Project affects a large number of user departments or units	somewhat
Demonstrate each new feature in an end-to-end use case. Develop a feature team that is organized around user-centered functionality. Each team is capable of delivering end-to-end value. Feature teams operate primarily with user stories, refactors, and spikes. Use capabilities as end-to-end solution services that support the achievement of user goals. Actively participate in ongoing agreements to maintain business and development alignment as priorities and scope are inevitably changed. Perform end-to-end and solution performance testing. Solution Demo demonstrate each objective and capability in an end-to-end use case. Capabilities are the end-to-end solution services that support the achievement of user goals. Implement capabilities via vertical, end-to-end slices of value, which enable incremental solution development.	
Project involves the use of new technology	definitely
Work with Agile Teams that perform research spikes, create proof of concepts, mock-ups, etc. PI Features are broken into stories and placed on team backlog. Local stories representing new functionality, refactors, defects, research spikes, and other technical debt are identified, written as enabler stories, estimated, and sequenced. Include enablers that could constitute infrastructure work, refactoring, research spikes, architectural improvement, and defects.	
Project involves use of technology that has not been used in prior projects	definitely
Work with Agile Teams that perform research spikes, create proof of concepts, mock-ups, etc. Ensure that the demo environments are adequate to the challenge of reliably demonstrating new solution functionality. Local stories representing new functionality, refactors, defects, research spikes, and other technical debt are identified, written as enabler stories, estimated, and sequenced. Include enablers that could constitute infrastructure work, refactoring, research spikes, architectural improvement, and defects.	
Project milestones not clearly defined	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Implement value stream coordination to ensure that the enterprise moves forward with each value stream in lockstep with the enterprise objectives. Align development to business via business context, vision, and Team and Program PI Objectives. Create a set of 'SMART' team PI objectives for each individual team with business value assigned. Business Owner prepares to communicate the business context, including milestones and significant external dependencies, such as those of suppliers. Define the roadmap, milestones, and releases. Perform PI planning to understand and agree on one or more iteration goal(s) based on the team and PI Objectives. Update progress toward Milestones, program PI objectives, and internal dependencies among the teams (by The RTE, Scrum Masters, and others (where appropriate)). Develop Program Increment (PI) objectives. Build Team PI Objectives. Build PI Objectives during PI Planning -ensuring the following is in place;. Differentiate between Features and Objectives. Facilitate trust among team members by having a "common mission, common iteration goals, and team Program Increment (PI) objectives". Write SMART Objectives (Specific, Measureable, Achievable, Realistic, Time-bound). (S) Provide a concise and simple description of the intended outcome (usually starting with an action verb). (M)easure what a team needs to do to achieve the team objectives (can be descriptive; yes or no; or quantitative; or within a range). (A) objectives should be achievable i.e; within the team's control and influence. (T)ime "period for achievement must be within the PI, and all objectives must be scoped appropriately".	
Project progress not monitored closely enough	definitely
Perform system demos. Perform system demo as near as possible to the end of the iteration. Demonstrate each new feature in an end-to-end use case. Summarize progress, feedback, and action items. Perform a final system demo of all features developed in the PI during the 'Inspect and Adapt workshop' at the PI boundary. Support the goals of synchronization, alignment, management of variability, and predictability of development velocity. Perform system demo after each system increment. Establish and communicate the annual calendars for iterations and Program increments. Facilitate PI planning readiness via fostering the preparation of vision and backlog, and via pre- and post-PI planning meetings. Actively circulate during planning, communicate business priorities to the teams, and maintain agreement and alignment among the stakeholders as to the key objectives of the train. Perform "demo as soon as the stories are ready". Measure the team's capacity based on previous sprint to establish the velocity for the upcoming iteration. Measure Velocity. Perform Team Demo at the end of each iteration. Calculate team" "derived velocity" by multiplying the throughput by an average story size".	
Team members lack specialized skills required by the project	somewhat
Involve a subject matter expert in basic exploration and sizing. Interact with analyst and subject matter experts during specification workshops. Work with stakeholders and subject matter experts to define the epic and its potential benefits. Interact with analysts and subject matter experts during specification workshops. Estimate stories based on Volume - (How much is there?), Complexity - (How hard is it?), Knowledge - (What's known?), and Uncertainty - (What's not known?). Adopt Communities of Practice to foster and support continuous learning. Team approve any scope, timing, or resource adjustments necessary to help ensure the release.	
Team members not familiar with the task(s) being automated	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Develop a feature team that is organized around user-centered functionality. Each team is capable of delivering end-to-end value. Feature teams operate primarily with user stories, refactors, and spikes. Refine the backlog. Involve with Agile team for short period of time. Attend iteration planning, backlog refinement meetings, system demos, and solution demos whenever critical UI-related work is involved. Perform backlog refinement. "Accept stories continuously to improve flow". Perform "demo as soon as the stories are ready". Refine the backlog before next planning to include the decision from demo and retrospective. Visualize the work. Use Conversation" throughout the story life cycle to include backlog refinement planning implementation and demonstration". Perform Daily Stand-up to understand team's status, "escalate problems, and get help from other team members". Perform backlog refinement. Define stories for team backlog.	
Quadrant 4: Environment	
Change in organizational management during the project	not at all
Corporate politics with negative effect on project	somewhat
Provide decision-making filters in the portfolio kanban, thereby influence the portfolio backlog. Define Strategic Themes. Formulate Strategic themes. Be aware that strategic themes could affect any of the major parameters including development/cycle time, product cost, product value, development expense, and risk. Provides the spending and personnel allocations necessary to accomplish the portfolio vision. Strategic themes provides direction. Participate, in some cases, in program portfolio management, product management, and even release management and system architecture. Shepherd the epics through the portfolio Kanban system and create the lightweight business case. Understand and communicate Strategic Themes and other key business drivers for architecture to system architects and nontechnical stakeholders.	
Dependency on outside suppliers	definitely
Business Owner prepares to communicate the business context, including milestones and significant external dependencies, such as those of suppliers. Work with Customers, stakeholders, and Suppliers to establish high-level Solution Intent; help establish the solution intent information models and documentation requirements. Work with Suppliers, making sure the requirements for supplier-delivered capabilities are understood, and assist with the conceptual integration of these concerns. Define Supplier responsibilities. Working with Lean-Agile Suppliers. Work with customers, stakeholders, and suppliers to establish high-level solution intent; help establish the solution intent information models and documentation requirements. Synchronize with Supplier and Solution Context. Solution Demo pulls various aspects of the solution together to ensure that the Agile Release Trains and Suppliers are creating an integrated and tested solution that is fit for its intended purpose. Allow Agile Release Trains and Suppliers in large Value Streams to build an aligned plan for the next Program Increment (PI). Perform pre-PI planning meeting to build the context that allows the ARTs and Suppliers to create their plans.	
Many external suppliers involved in the development project	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
. Business Owner prepares to communicate the business context, including milestones and significant external dependencies, such as those of suppliers. Work with Customers, stakeholders, and Suppliers to establish high-level Solution Intent; help establish the solution intent information models and documentation requirements. Work with Suppliers, making sure the requirements for supplier-delivered capabilities are understood, and assist with the conceptual integration of these concerns. Define Supplier responsibilities. Working with Lean-Agile Suppliers. Work with customers, stakeholders, and suppliers to establish high-level solution intent; help establish the solution intent information models and documentation requirements. Synchronize with Supplier and Solution Context. Solution Demo pulls various aspects of the solution together to ensure that the Agile Release Trains and Suppliers are creating an integrated and tested solution that is fit for its intended purpose. Allow Agile Release Trains and Suppliers in large Value Streams to build an aligned plan for the next Program Increment (PI). Perform pre-PI planning meeting to build the context that allows the ARTs and Suppliers to create their plans.	
new: Country-specific regulations	definitely
File patent, certify the system, audit certain regulatory requirements for economic success of product development. Defines how the system builder's system intent must be organized, packaged, and integrated for use by the Customer to meet any compliance, certification, and other objectives. Supports compliance and contractual obligations. Use Traceability for Impact Analysis and Compliance. Simplifies and automates most regulatory and contractual compliance needs. Generate documents for regulatory compliance (FAA, FDA, etc.) or contractual obligations (CDRLs in government contracting) when required. Collaborate (System Engineers) with Customers and/or regulatory agencies on the minimum set sufficient to meet their obligations. Invest in solution intent documentation in more complex and/or regulated environment. Establish the solution intent's organizational structure and define where various types of information are managed to support analysis and compliance needs. Record what is needed because solution intent is the means to the end of building a product and meeting compliance and contractual obligation.	
new: Delays caused by global distance	somewhat
Establish high-bandwidth communication across all team members and stakeholders. Calculate the Cost of Delay. Manage and optimize the flow of value through the program using various tools, such as the Program and Value stream Kanbans and information radiators. Manage and optimize the flow of value through the program using various tools, such as the Program and Value Stream Kanbans and information radiators. Use big visible information radiator (BVIR) on a wall in the team room. Constant communication. Use webcams, instant messaging, and other collaboration tools if teams are distributed. Use 'big visual information radiator' (BVIRs) to understand and track progress during iteration execution. Consider upgrading and enhancing "intra- and inter-team communications systems". Use these bands and buffers to allow teams extra time to respond to unforeseen events, delays in dependencies, and other issues. PI planning, iteration planning, backlog refinement, inspect and adapt, architecture discussion, etc., all benefits from frequent meetings [Schedule frequent meetings using a predictable cadence]. Perform constant communication and collaboration Collaborate with large stakeholder community to determine the best course of action. Invest in integration, testing, and supporting infrastructure to integrate big systems. Facilitates cross-discipline collaboration by allowing traceability from a model in one discipline to a model in another. Collaborate (System Engineers) with Customers and/or regulatory agencies on the minimum set sufficient to meet their obligations.	
new: Lack of architecture-organization alignment	somewhat

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Identify dependencies and forecast cross-team and cross-ART coordination. Create enabler features by architects or engineers to pave the architectural runway which is maintained in the program backlog alongside business features. Create architectural enablers to pave the runway. Organize planning enabler such that the system can run for most of the time on the old architecture or infrastructure. Intentional Architecture Support the Bigger Picture. Provide guidance for cross-team design and implementation synchronization. Enable Flow and Agility with Architecture. Split enabler epics into enabler features and/or capabilities, which are ultimately implemented by individual ARTs. Optimize subsystem ARTs for architectural robustness, critical components, or components that are used by many other elements. Enablers provide for exploration of new capabilities, contribute to solution infrastructure and architecture, and enhance NFRs. Continuous evolution of the architectural runway support current and near-term features. Business Owner understands and assures that business objectives are understood and agreed to by key stakeholders of the train, including the Release Train Engineer (RTE), Product Management, and System Architects. Understand and communicate Strategic Themes and other key business drivers for architecture to system architects and nontechnical stakeholders. Work with portfolio stakeholders, particularly the Enterprise Architect, to develop, analyze, split, and realize the implementation of enabler Epics. System and Solution Architect/Engineer works with Product and Solution Management to determine capacity allocation for enablement work. Work with system and solution architect/engineering and teams to build and maintain the technical foundation for real-time user experience validation, feedback, tracking statistics, etc. System and Solution Architect /Engineer participates in planning, definition, and high-level design of the solution and explore solution alternatives. System and Solution Architect /Engineer works with product and solution management to determine capacity allocation for enablement work. Develop enabler stories to "support exploration, architecture, or infrastructure". Manage dependencies and resolve impediments by continuously and actively engaging with other teams. Include enablers that could constitute infrastructure work, refactoring, research spikes, architectural improvement, and defects. Solution intent records and communicates requirements, design, and system architecture decisions. Enablers provide for exploration of new capabilities, contribute to solution infrastructure and architecture, and enhance NFRs. That drives early value delivery and architectural robustness. Generate document for stakeholders with different system perspectives or architectural framework standards (e.g., DoDAF, MODAF) that defines multiple stakeholder viewpoints.	
new: Lack of face-to-face interaction inhibits knowledge sharing	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Provide guidance for cross-team design and implementation synchronization. Manage and optimize the flow of value through the program using various tools, such as the Program and Value stream Kanbans and information radiators. RTE facilitate periodic synchronization meetings, including the ART sync at the Program level and the Value Stream (VS) sync at the value stream level. Encourage team level, program level, and value stream level continuous integration and community of practices around SAFe, Agile and Lean. And, around engineering and quality practices. Business Owner attends occasional Agile team iteration planning and iteration retrospective meetings, as appropriate. Manage and optimize the flow of value through the program using various tools, such as the Program and Value Stream Kanbans and information radiators. Work with Customers, stakeholders, and Suppliers to establish high-level Solution Intent; help establish the solution intent information models and documentation requirements. Participates in the PI Planning and Pre- and Post-PI Planning meetings, where they present what they plan to deliver in the next Program Increment, along with an indication of what will be delivered in each Iteration. Lead UI design and User experience/UI community of practice workshop. Attend iteration planning, backlog refinement meetings, system demos, and solution demos whenever critical UI-related work is involved. Work with customers, stakeholders, and suppliers to establish high-level solution intent; help establish the solution intent information models and documentation requirements. Visualize the status of the stories, defects, and other activities that the team is working on during the iteration. Use big visible information radiator (BVIR) on a wall in the team room. Use "Agile Project Management Tools to capture stories and status, defects, test cases, estimates, actuals, assignments, burn-down chart". Use webcams, instant messaging, and other collaboration tools if teams are distributed. Use 'big visual information radiator' (BVIRs) to understand and track progress during iteration execution. Perform daily Stand-up meetings for team coordination. Provide governance for any upcoming Releases and also provide regular communication to management. PI planning, iteration planning, backlog refinement, inspect and adapt, architecture discussion, etc., all benefits from frequent meetings [Schedule frequent meetings using a predictable cadence]. Create a "Design Community of Practice (CoP)". Team learn together and share best practices through inter-team communication supported by Communities of Practice (CoP). "Business owner assign business value to each of the teams individual objectives in face-to-face conversation with the teams". Collaborate with large stakeholder community to determine the best course of action. Bring together stakeholders from all parts of the value stream. Solution intent records and communicates requirements, design, and system architecture decisions. Adopt Model-Based Systems Engineering that provides an effective way of reasoning about the solution and also serves as an efficient communication tool for sharing this knowledge. Encourages general knowledge discovery by making information, and related cross-discipline information, more accessible to teams. Store information into repositories that can be used for inspections and formal reviews. Establish the solution intent's organizational structure and define where various types of information are managed to support analysis and compliance needs. Participate in the creation, feedback, and refinement of solution intent information. Record what is needed because solution intent is the means to the end of building a product and meeting compliance and contractual obligation.	
new: Lack of process alignment	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Adopt decentralized, program, and team-based rolling-wave planning via routine and cadence-based PI Planning activity. Program Portfolio Management (PPM) team continuously assesses and improves their processes using a structured, periodic Self-Assessment. Perform a structured, root cause analysis-based, problem-solving workshop for large, systematic program-level problems. Brainstorm solution. Release on the program increment cadence. Develop on Cadence, Release Any Time. Support the goals of synchronization, alignment, management of variability, and predictability of development velocity. Participate in Inspect and Adapt to improve the release process, value stream productivity, and solution quality. Team evaluates its process and any improvement stories it had from previous iteration. Use "iteration retrospective to drive program level changes to process either immediately or in the Inspect and Adapt workshop". "Management alignment and organizational readiness for planning". "Allows Agile Release Trains and Suppliers in large value streams to build an aligned plan for the next program increment". Develop Cadence Principles. Apply Cadence Principles. Planning at regular Program Increment (PI) intervals limits variances to a single PI time-box, thereby increasing Agile Release Train and Value Stream predictability [Use a regular cadence to limit the accumulation of variance]. PI planning, iteration planning, backlog refinement, inspect and adapt, architecture discussion, etc., all benefits from frequent meetings [Schedule frequent meetings using a predictable cadence]. Apply Synchronization Principles. Individual Agile Teams are aligned to common iteration lengths [Exploit economic of scale by synchronizing work from multiple projects]. Value Stream and Program PI events synchronize customer feedback, resource and budget adjustments [Use synchronized events to facilitate cross-functional trade-offs]. Teams are aligned to common time-boxes and similar batch sizes [To reduce queues, synchronize the batch size and timing of adjacent processes]. Teams integrate and evaluate (at least) on iteration boundaries; program and value streams integrate and evaluate on PI boundaries. [Apply nested cadence harmonic multiples to synchronize work]. Develop iteration goals to align team members to a common purpose. Develop iteration goals to align program teams to common PI Objectives and Manage dependencies. Common Cadence. Use stretch objectives provide capacity margins required to synchronize to a delivery cadence. Allow Agile Release Trains and Suppliers in large Value Streams to build an aligned plan for the next Program Increment (PI). Track changes as both the system and deployment environment have to evolve to a common state. Perform continuous Integration and testing while Solution Demonstration occurs on a fixed PI cadence.	
new: Lack of tool/infrastructure alignment	definitely

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Manage dependencies by applying extensive degree of cooperation; a common value stream backlog; implementation of new, crosscutting capabilities; additional system integration; additional roles and responsibilities; special considerations for pre-, post-, and PI planning activities; different degree and types of DevOps support. Create infrastructure enablers to be ready to develop, test, and integrate the initiatives. Develop infrastructure in different levels of SAFe to support frequent or continuous integration and testing. Integrate a subset of capabilities, components, or subsystems. Define solution integration, testing, and demo. Adopt continuous integration and testing in order to progress configuration management, automation, and virtualization. Apply Continuous Integration practices with test automation whenever feasible to monitor and ensure progress. Manage and optimize the flow of value through the program using various tools, such as the Program and Value stream Kanbans and information radiators. Encourage team level, program level, and value stream level continuous integration and community of practices around SAFe, Agile and Lean. And, around engineering and quality practices. Manage and optimize the flow of value through the program using various tools, such as the Program and Value Stream Kanbans and information radiators. Encourage Team Level, program level, and value stream level Continuous Integration and Communities of Practice around SAFe, Agile, and Lean and around Engineering and Quality Practices. Create and maintain infrastructure, including continuous integration, automated builds, and automated build verification testing. Run solution-level integration scripts or integrate manually where automation is not possible or has not yet been applied. Use "Agile Project Management Tools to capture stories and status, defects, test cases, estimates, actuals, assignments, burn-down chart". Use webcams, instant messaging, and other collaboration tools if teams are distributed. Adopt "automated testing to quickly perform regression testing, enhancing continuous system-wide integration, refactoring, and maintenance". Understand requirements for working on "technical infrastructure, tooling, and other systemic impediments". Demo these integrations at the system demo at the end of every iteration. Use the IP iteration as a placeholder for full, final solution integration that must happen at least once per PI. Erect new continuous integration environments within the development infrastructure. Adopt "project management tooling". Consider upgrading and enhancing "intra- and inter-team communications systems". Detail what the ART will have ready for integration and demo at the end of the PI. Team approve any scope, timing, or resource adjustments necessary to help ensure the release. Ensure that integration between teams on the same ART occurs on a regular basis and no less than every iteration. Perform Continuous Integration. Implement feature and component level continuous integration. Implement ART integration. Solution Integration. Enable continuous integration. Infrastructure. Engineering Techniques in Support of CI. Make Continuous Integration a Culture. Make integration results visible. Make fixing a failing integration top priority. 'try for early and frequent integration and testing of subsystems and systems' including hardware/firmware. Invest in integration, testing, and supporting infrastructure to integrate big systems. Leverage virtualization, environment emulation, mocks, stubs, reduced test suites, etc to assist integration and testing. Allocate time and effort for integration and demonstrations during PI planning. Track changes as both the system and deployment environment have to evolve to a common state. Adopt Model-Based Systems Engineering that provides an effective way of reasoning about the solution and also serves as an efficient communication tool for sharing this knowledge. Solution Integration, Testing, and Demo. Perform continuous Integration and testing while Solution Demonstration occurs on a fixed PI cadence.	
new: Unstable country/regional political/economic environment	somewhat
Build an economically viable solution. File patent, certify the system, audit certain regulatory requirements for economic success of product development. Require evolution of the Customer's deployment environment for new solutions. Track changes as both the system and deployment environment have to evolve to a common state. Invest in solution intent documentation in more complex and/or regulated environment.	
Organization undergoing restructuring during the project	not at all
Resources shifted from the project due to changes in organizational priorities	somewhat

Continued on next page

Table 4 – continued from previous page

GSD Risk Catalog risk/Practices	*Level*
Apply WSJF to prioritize the items in the Value stream and Program backlogs. PI cadence offers time box experimenting approach which later help to produce the desired and long term results. Apply WSJF prioritization to compare business and enabler epics against each other. Apply WSJF for continuous prioritization of features in the program backlog. Use program backlog prioritization via WSJF. Develop on Cadence, Release Any Time. Perform initial exploration of epics and rank them roughly by using Weighted Shortest Job First (WSJF) to determine which epics should move to the next step for deeper exploration. Elaborate and approve highest-priority features then move to program backlog where they are prioritized with WSJF relative to the rest of the backlog. Apply Weighted Shortest Job First prioritization method for job sequencing. Define Weighted Shortest Job First (WSJF). Calculate the Cost of Delay. Epic Owner works with development team to size the epic and provide input for economic prioritization based on WSJF. Participates in the PI Planning and Pre- and Post-PI Planning meetings, where they present what they plan to deliver in the next Program Increment, along with an indication of what will be delivered in each Iteration. Planning at regular Program Increment (PI) intervals limits variances to a single PI time-box, thereby increasing Agile Release Train and Value Stream predictability [Use a regular cadence to limit the accumulation of variance]. If a feature doesn't make it into a PI (or release) and it remains high priority, its delivery can be anticipated to be on schedule in the next PI [Use cadence to make waiting time predictable]. Use either size/value or Weighted Shortest Job First (WSJF) to prioritize the stories issued by product owner. Use stretch objectives provide capacity margins required to synchronize to a delivery cadence. Pull jobs into implementation based on WSJF, where estimate of job size is typically used as a proxy for duration. Apply WSJF prioritization to sequences the features and capabilities that can be pulled into the backlog.	
Unstable organizational environment	not at all

4 Risks Seen in Cases

As part of two longitudinal case studies of two companies engaged in scaling agile development, we conducted interviews, observed ceremonies, [4, 5] and administered self-assessment surveys [10]. In this first step, we examined these data for evidence that the companies had experienced problems (or not) related to the risks in our GSD Risk Catalog.

Then, in the second step, we assessed the frequency at which each company performed the respective scaling agile framework practices mapped to risks in the GSD Risk Catalog.

Finally, in the last step we connected the output of the previous two steps, to understand whether the scaling agile practices eliminated or mitigated the corresponding risks: if the practices were implemented, and no evidence of the risk was seen, the practices could have been material in *eliminating* the risk. If the practices were implemented in the company, but there was evidence that the risk was a problem for the case, the practices still might have been effective at *mitigating* the risk; or, this might indicate that the theoretical mapping is not effective in practice.

To determine which of these alternatives was the case, we considered three additional elements:

1. *Strength of theoretical mapping:* the degree to which the practices address the risk. Risks that are only "somewhat" addressed (by the practice), are perhaps more likely to be seen as problems.
2. *Strength of practice implementation in cases:* the frequency at which the associated practices were performed. If this was less than "always," it's possible the practices were not effective because they were not thoroughly implemented.
3. *Level of control:* whether the risk can be *eliminated*, or only *mitigated.* Certain risks, such as *Unstable country/regional political/economic environment*, are part of the environment; they cannot be eliminated, but their impact can be reduced.

We present the results of applying this method are presented in Table 5 and Table 6.

4.1 Risks Seen in Case A

Table 5: Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Quadrant 1: Customer Mandate
Conflict between users
PA1: "Because they've [product managers] been in the US and I've been here. There's also that product is so complex, the knowledge in the domain makes it has some barriers, having your peers respect your opinion on things. Because no matter what, you're thinking, if you haven't had ideas or experience, it can be so much harder for them to hear your ideas. with UX, there're so many guiding principles that are sort of failing universally, studying the users, making things simpler, don't bother them with a lot stuffs all the time. That's sort of things that people could not appreciate and they wouldn't even hear it because they know the product, it works like this, and they assumed the customers wanted the way it was."
Lack of cooperation from users
(not observed in case)
Lack of top management support for the project
(not observed in case)
Lack of user participation
(not observed in case)
Lack or loss of organizational commitment to the project
(not observed in case)
Users not committed to the project
(not observed in case)
Users resistant to change
(not observed in case)
Users with negative attitudes toward the project
(not observed in case)
Quadrant 2: Scope and Requirements
Conflicting system requirements
PA1: "Sometime the user objectives and business objectives are at odds, and you've got to find a way to make it work for both, sometimes you're implementing a business objectives that a customer/user doesn't like. Sometimes the business objectives are the primary things, not the UX."
PA1: "The directions were, it's going to work on the iPad, make it as simple as possible... So, we did some great brainstorming sessions, there were a lot of fantastic ideas around that sort of thing, but then when we started on the project, stakeholder (the product management and POs) had different point of view, and it sort of switched from simplest possible to can you please translate what we have in the desktop app into the web."
Continually changing project scope/objectives

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
(not observed in case)
Continually changing system requirements
(not observed in case)
Difficulty in defining the inputs and outputs of the system
PA1: "Before we had multiple features being built at the same time across the team, and the design work is done within the same time frame as the development, so it was very difficult to ask business to allocate the time to do design, we never got time to do research. It was a bit of a rush. We only had me and one other UX person, we might have 8 features, that's probably the worst, and it can be very complex."
Ill-defined project goals
(not observed in case)
Incorrect system requirements
(not observed in case)
System requirements not adequately identified
PA1: "We used to use a use case supported by journey, but now we're using user stories. The user stories that we're writing now is an overall of the thing the user wants to do, but written from his/her perspective. The goals are saying, this is the thing that user needs, then we have conversation on ways to deliver that, like what solution matches that, iterate through that and find the right way through."
PA2: "everything has to be user stories, sometimes we think that's the underlined problem ... we get too focused on that because while it's good to have those stories to get you going ... the magic happens every day in the team making adjustments, embracing that, rather than trying to design user stories to end."
PA8: "No, we're not. Here, there are aspects of a work here that is agile and aspects of our work that's not agile, and there's a large communication bottleneck. Or non-collaborative thing... Probably what you would the concept of product and release plan, although there have been significant changes recently we are not involved with it."
Unclear system requirements
(not observed in case)
Undefined project success criteria
(not observed in case)
Users lack understanding of system capabilities and limitations
(not observed in case)
Quadrant 3: Execution
Development team unfamiliar with selected development tools
PA2: "We have a history of reinventing the wheel ... at what's out there that's going to make life easier for us to work, what problems already been solved."
Frequent conflicts among development team members
(not observed in case)

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Frequent turnover within the project team
(not observed in case)
High level of technical complexity
PA2: "Prioritising technical feature, we had the task of improving the performance in cloud solutions, no PO or PM know what to do."
PA2: "Supposed to code less, supposed to mentor more, work more with others. The complexity of some of the things that we architecture owner, staff engineers dealt with, we end up having to do a lot of disproportionate amount of coding because of the complexity."
PA2: "The thing that we have to deal with is that we have a lot of legacy code that was already built in those 3 tiers, and we brought a lot of that across . . . we were sort of forced to stay with it because the need to reuse our code, we just didn't have time to build everything. . . lot of room for improvement I think, if we started from scratch I think things will be a bit different."
Highly complex task being automated
PA2: "[We] have about five sort of databases, each hosting a number of tenants, at least two different geographies, US and Europe. I think our biggest database has got like 50 tenants in it, and the other are starting to fill up as well, and we've reached the limit before, and we just simply start up another server. It is not a simple matter of one server, you have to, it's all clustered machines, you have to through 3-4 different servers you have to bring up to every new database."
Immature technology
PA2: "We're also not shy building our own tools, building our own code generators, things for specific problems, having ten people writing million lines of codes, which is also been our historical problems."
Inadequate estimation of project budget
(not observed in case)
Inadequate estimation of project schedule
(not observed in case)
Inadequate estimation of required resources
PA4: "It's been years of reviewing, giving feedback, being overall it's going out, shortcut, running late, all over the walls with bugs, you know, you feel like you're trying to help, but there's no time for you. So, that's been the bulk of the time, and now that the DAD process is coming I'm hoping that we can now incrementally improve on what we already have, and not introduce more problems."
Inadequately trained development team members
PA2: ". . . we struggle with small customers, where they can't afford, or we can't to give them consultant or help, so we really haven't put enough effort in self-service, making it easy to use, a customer can't just sign up and get a trial of the software, they can't just sign up and pay on the credit card and start using it. It's complete opposite of that, we need many people involved in [product], in order to have any customer start using our software."
PA2: "We did not get any training, we're struggling with it, because it's just a complete different way of thinking for us. And we have some people naturally fighting against it . . . obviously gonna be more frustrating for the more experienced person who could probably work 2-3 times faster, but it's really about the education, it's what we're trying to achieve. In theory, hopefully productivity and quality increases as well, two minds on the job, but it's been a struggle."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Ineffective communication
PA2: "I've seen problems where people gets too Jira focused, and they stopped talking. I think there's still a lot of value in us talking to each other, and Jira has sometimes taken that away. Obviously you've got to find the balance."
Ineffective project manager
(not observed in case)
Inexperienced project manager
(not observed in case)
Inexperienced team members
(not observed in case)
Lack of an effective project management methodology
PA6: "Even though provide 100% automation for cloud and on-prem, there's always the particular failure, in that scenarios that's probably harder to code. So there's component of manual testing and also exploratory testing that we do. Automation has played a bigger role for us, we reduced the release cycle from 6 weeks to 2 weeks, without automation to give us that nightly check ability, it's very difficult, and our product is very complex, just setting up the environment and do manual testing is weeks of effort."
PA6: "We got a definition of done, in that two weeks period, it has to be coded, documented, and it's got to be QA. So nothing gets released in those two weeks, if it's not QA. And also, in the criteria being done, it's as close as possible to 100% automation... it's something we didn't have before. We used to concentrate on a lot of manual work, now we say to the team, it's just your responsibility as QA to automate."
Lack of commitment to the project among development team members
(not observed in case)
Lack of people skills in project leadership
(not observed in case)
Large number of links to other systems required
(not observed in case)
Negative attitudes by development team
PA2: "Actually in some ways, it's even negatively impacted that. Because of our complex environment, cause we have people that know area so well, and DAD pushes generalist, so it's been a tug of war. We're going into our dashboard project, heavy database stuff, and we don't have much database experience in the team, and we have to deal with that. I would love to have database team, maybe you could do that with DAD, but we haven't. Maybe that's why we need the education so bad."
PA2: "We didn't make that decision, it was made for us. But having said that, we still have a degree of team independence, being able to decide how we do things. Some things are mandatory, sort [of] 50-50, some things are mandated, but we do get to make decisions."
new: Ineffective collaboration

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA1: "The directions were, it's going to work on the iPad, make it as simple as possible, the least interaction, the most results, the users want to see progress towards certain goals, and have me make that visible and simple. So, we did some great brainstorming sessions, there were a lot of fantastic ideas around that sort of thing, but then when we started on the project, stakeholder (the product management and POs) had different point of view, and it sort of switched from simplest possible to can you please translate what we have in the desktop app into the web."
PA2: "I think mostly it works well, I've seen problems where people gets too Jira focused, and they stopped talking. I think there's still a lot of value in us talking to each other, and Jira has sometimes take that away."
PA2: "We've been using that for years, everything has to be user stories, sometimes we think that's the underlined problem, as in not doing enough of those in a thorough way, getting the right people to do those is the cause of a lot of our challenges. I personally think that sometimes we get too focused on that because while it's good to have those stories to get you going, I still think that, most of the work, how you gonna do this, what makes sense is in the future, everyday there's a decision point. We might need to change directions slightly, and sometimes you end up with this concept you can design everything upfront, there's a user stories for everything, and you can just get people to build it, no one has an original thought, to me that makes crap software. While you do need a direction to start with, really the magic happens every day in the team making adjustments, embracing that, rather than trying to design that out, like designing user stories to end."
PA3: "The area that is less 'T' skilled is that component , and that requires a lot of depth and knowledge about how that works. So, one team is generally good at doing that, while the other teams know it, but they don't exactly hang together."
PA7: "It does happen from time to time, generally at the end of the iteration, we would have a retrospective with the team, and those retrospective would analyze, effectively come up with a root cause analysis, so why something fall out, because we didn't understand it, why didn't you understand it, because I t wasn't clear about the goals, why weren't the goals clear, so we always go and ask why, to get to the cause."
PA7: "The team leads need to know the how, how things are hanging together, the formal HR management stuff, to some degree, linked. But we're specifically decoupled, because in a team, we want your team members at ease to discuss something with the team lead, not seeing the team lead as a boss, if they do, there's a level of formality and that then may restrict the flow of information, not what we want."
PA7: "Typically, we found that, although our goal is to keep it in 15 minutes, if it goes on for another minute or two, it's not the end of the world. If it goes on for much more than that, we'll look at the structure of what we're trying to achieve, it's often the more expensive thing that extends the meeting about 3-4 minutes, we like to cover everyone and keep it to that time."
PA8: "Basically the job of the PO to choose the priority for the most part. But you also end up with a lot situation where if it gets too technical, the developers has to choose the priority, usually the AO has to do that. It is a collective effort, it is usually PO has got this stories but he/she just doesn't know enough to know what's the sequence of this in order to get it to delivery, and that's usually where I would say no, this is the first thing we need to do, and this is the highest risk, and then we'll do this. Usually verbal, and magically Jira gets updates by somebody, probably the PO, and suddenly that's what the team are working on."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA8: "The question remains, what's the most effective team size, certainly it's bigger than what we used to have, I'm not sure if it's bigger or smaller than we currently have, but we're doing an experiment with a team of 7, that just starter now, 4 teams, 3 areas about 12-13 engineers, one team is 7 but that's experimental."
new: Ineffective coordination
PA3: "Basically in our team the QA people are team member just like the developers, so very closely, some QA people have coding experience so we can give them those tasks, some of them don't so we can't do that. But they're equal team member."
PA3: "Currently we've got 4 team, 3 of those teams are large [12-13 indviduals], one team which is currently at 6-7 people. In theory, the stand-up meeting would be shorter, but we're a new team, so we're still in the forming stage, haven't got that efficiency yet."
PA7: "Business verification, yes. we have some other verification tasks which would not that our PO's responsibility. It may not always fall on the PO to do business verification, the way we work in our team is generally that way."
PA7: "differences between the roles within the team, so everyone's equal and everyone's contributions are valuable, and those roles give you responsibility to do certain thing in addition to your team responsibilities, so if you go from functional manager, managing team and doing all the leadership role effectively, now only 1/3 of the leadership role in the new agile environment. So, that has been the challenges, because you typically, automatically try and cover for those other 2 roles that you no longer do, somebody else is doing that, step back form that, that's how the new structure let them get on with their role."
PA7: "if someone goes on leave, you just slow down so much. With the agile environment, with the large team of 12 people, one person takes leave, that's less than 10% of your lost velocity, you find that there's always somebody on leave."
PA7: "In the new model, we have embedded QA, documentation, but we effectively have embedded all 3, the embedded UX is less of a distinct category, because I'm guessing UX roles were naturally integrated, the people are still developers. We have embedded documentation writer in the team, but their skill set is so different to the developer that they generally just do documentation, although we would encourage them to be 'T' skilled, so that they could be good at one specific and broader along everything rather than just one specialty."
PA7: "The technical writers, they've got an eye for usability, but our workflow and so on often modeled by our PO and that has a separate task or role to documentation. I don't think all of our team have a doc writer, I think the smaller one doesn't, so that is a challenge because now the doc writer has to cover for that team, or they have to do it themselves. The doc writer did raise concerns of usability; the focus was on documentation."
PA7: "what we found is the behavior you need to exhibit in the leadership is different to how you do in the manager role. You need to actually learn to step back from trying to have the top down approach, instead of pushing request down, you need to allow the team to step up and bottom up, encourage that approach within yourself and to everyone."
PA7: "when you encounter a support ticket or some other work that's out of line of what your direction and goals are, the impact on your velocity is greater, suddenly you have to do hot fix for production you didn't know about until yesterday, it's gonna take about 2 weeks, effectively half of your team is out for 2 weeks, and your progress is halted."
new: Lack of trust

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA1: "We have items in the roadmap that may never come into project, around being able to put logo, change some basic color thing."
PA2: "I guess we still even today, we sort of follow fairly traditional 3 tier, presentation, business logic, and data link. Pretty much stuck to that, even though we moved to the web have changed things a bit. Basically our code is still structured based on those 3 layers."
PA2: "I'm resource driven, like I'm not comfortable until I see a path through to actually a working piece of software. My guidelines are always work at the highest priority things, because there's no point in having 10 priorities, have 3 or 1, and just work that out, work at what is the biggest risk, work at what is the biggest problem and solve it. Solve the main thing, and not worry about the little thing, just cut them off nicely. Do the main thing, make it work really well."
PA2: "I think there's big improvement we could make if we went greenfield. You might end up with those 3 tiers, but there's technology to actually help you so that the whole data is not really a big deal anymore cause you've got all tools to help you with that kind of stuff. Certainly, lot of room for improvement I think, if we started from scratch I think things will be a bit different."
PA2: "Pair programming, we did not get any training, we're struggling with it, because it's just a complete different way of thinking for us. And we have some people naturally fighting against it. It's quite an effort for us, it's obviously gonna be more frustrating for the more experienced person who could probably work 2-3 times faster."
PA2: "That's a really major challenge. So we have to come up with some really crazy ideas in order to improve performance."
PA2: "We do have our own cloud platform, it's been a struggle... always had small cloud platform, and when we moved our product to it, it was such complex, huge, massive product, so much more so than the other products. So it was a learning for us, because we've never been on the web or cloud before. But also big learning curve for our Ops people, who are hosting all these server."
PA2: "We do, we have challenges, different time zones, way of thinking, and we tend to have a handful of people who developed good relationships with them, without that it's really difficult."
PA3: "In DAD, the theory is that any team can do anything, so you've got maybe teams playing in the same space at the same time, We still keep competencies of the team, but it's not as strong as defined as it used to be (squad), I think we lost something there."
PA3: "I wouldn't call it so much, I think a lot of software engineers prefer to work alone, it's actually a struggle to them to work together, you just want to be like, leave me alone for three days and I'm going to come up with something amazing."
PA3: "Now it's more about generalization, I'm sure people have been talking about 'T' skills. We've got the generalization and specialization, but It does feel like we are towards generalization more than specialization."
PA3: "To me, ownership was the important thing that we got from it. So, the focus was very much on an area of the product, this is something that you will get to know and be expert in, contrasting that, what we're doing that now, the focus isn't so much on the area of the product, it is more of a time-based focus. This is the iteration, we've got this to deliver, let's pluck out all other distraction and just focus on our tasks."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA5: "DAD doesn't really deal with buffers in that same way. What it does instead, it says hey, slow down, what did you do during your day? You're an AO, so you end up mentoring and going to all these meetings, roughly how much is that time? If you start adding up all that overhead, what you do instead, you work out what your capacity is, individually and across the team.Once you take away all those overheads, there's not so much buffering the task, as in reducing your capacity to do those tasks."
PA5: "On the cloud, you don't have that safety at all, 100% responsible of the security about cloud solution. It has been a big learning curve, we did not have a lot of experience in security years ago, and we had to learn sometime the hard way. Recently we had to roll out some security vulnerable fixes. You don't enjoy doing that, you have to sort of swallow your pride."
PA5: "Sometimes depends on how complex the area is, it really was a matter of me basically just working out the steps and saying I gotta do this one, you gotta do this one, really was me just handing out the tasks to do, because I was sort of the only who has the big picture of what and how we're gonna do this."
PA5: "We've all been going into areas we don't have a lot of expertise, like web programming, it wasn't our forte 10 years ago. And now we definitely have leveraged a lot open source for that kind of thing. We're also not shy building our own tools, building our own code generators, things for specific problems, like having 10 people writing million lines of codes, which is also been our historical problem."
PA7: "We have a release checklist, a whole bunch of tasks there, it's not as well tied to the iteration as what I'd like. we're not quite there yet, so those tasks will be assigned to typically one of the leadership people, but not always the case."
PA8: "Actually in some ways, it's even negatively impacted that. Because of our complex environment, cause we have people that know area so well, and DAD pushes generalist (T-skilled), so it's been a tug of war. We're going into our dashboard project, heavy database stuff, and we don't have much database experience in the team, and we have to deal with that. I would love to have database team, maybe you could do that with DAD, but we haven't. Maybe that's why we need the education so bad."
PA8: "We did have conversation recently about measuring the numbers of refactoring in iteration, recommended by agile coach. I didn't really like the idea, because comparing one refactor to another is not equal. I don't think there are any rules about it, whatever people are comfortable with."
PA8: "we struggle with small customers, where they can't afford, or we can't to give them consultant or help, so we really haven't put enough effort in self-service, making it easy to use, a customer can't just sign up and get a trial of the software, they can't just sign up and pay on the credit card and start using it. It's complete opposite of that, we need many people involved in order to have any customer start using our software."
One of the largest projects attempted by the organization
(not observed in case)
Poor project planning
(not observed in case)
Project affects a large number of user departments or units
PA2: "We do, we have challenges, different time zones, way of thinking, and we tend to have a handful of people who developed good relationships with them, without that it's really difficult."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA6: "On-premise, you might break one customer. If you introduce a defect on the cloud, you might break 20 or 50. Sometime the impact that it has on the cloud is wider."
Project involves the use of new technology
PA2: "... every release, there's one or two items that had to do with other technology, some new tools, some API, some metadata they got, that we need to consume and fit into our model."
Project involves use of technology that has not been used in prior projects
(not observed in case)
Project milestones not clearly defined
(not observed in case)
Project progress not monitored closely enough
PA6: "Yes, we still find even, once we get to the final IR, and we've gone to those two weeks getting ready to release to production."
Team members lack specialized skills required by the project
PA1: "There's a lot of different things that affect what you can and can't do. Whenever we need to make a change to a component, it's a bigger piece of work, there's fewer people that are capable of doing the implementation, so we've got the 40 or so people, but only 1-2 who can change them. So, that tends to limit the solutions, you work with the kinds of filters and search techniques that you already have in your components, rather than reinventing them every time. That's probably the biggest limitation we have in terms of the technical."
PA2: "it's been a learning curve for us, because we came from desktop environment which is safely within an intranet, security was obviously important, but it was like 50-50, it was their security as much as it as ours, as long as we don't do anything too stupid. On the cloud, you don't have that safety at all. So we're responsible of the security about cloud solution. It has been a big learning curve, we did not have a lot of experience in security years ago, and we had to learn sometime the hard way. Recently we had to roll out some security vulnerable fixes. You don't enjoy doing that, you have to sort of swallow your pride."
PA2: "To be honest, there are very few people here, if you put them in a room, gave them a week or two, they would be able to produce something that we can release. There'd be some big area that just wouldn't work, like performances."
PA2: "We've all been going into areas we don't have a lot of expertise, like web programming, it wasn't our forte 10 years ago. And now we definitely have leveraged a lot open source for that kind of thing."
Team members not familiar with the task(s) being automated
PA2: "Some things are more straightforward and there's no really architecture. We certainly look at the risks, like if this going to perform well and we tried to do all that front work. When you got something like what we're doing right now, rolling out a new dashboard, massive architectural spikes at the start, cause we got to work it out, how to put the data we have, massive amount of data, how to summarize it, and everything have to perform well ... bring that kind of stuff to the architecture upfront."
Quadrant 4: Environment
Change in organizational management during the project

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
(not observed in case)
Corporate politics with negative effect on project
(not observed in case)
Dependency on outside suppliers
(not observed in case)
Many external suppliers involved in the development project
(not observed in case)
new: Country-specific regulations
PA3: "Regulations worldwide, Normally it's done by the Product Managers, information is given to us, most of the teams, each team will have POs working with a product mananger for a particular countery, and so data just flows, from them into the teams."
PA3: "we can't do much else, because we just got rid of all the access to the database and also limited access to the file system. The reason is because of the sensitivity of data, even if they give a copy of the database which we don't have locally, we have to normalized and make sure we removed any information that is related to the customers, even then, the database cannot leave the data center. Get the copy, change the data, and keep the shape, massive procedures. For Europe, completely different story, can't even leave the European countries."
PA3: "We do linguistic as well, we have to support multiple languages, that's another thing we have to cover."
PA8: "Yes, the PO were meant to create storyboard for new features, for you to be able to create a storyboard, you've got to be in contact with the Product Management and with sales and service to make sure whatever we produce is suitable for the market and meet customers' requirements."
new: Delays caused by global distance
PA3: "Going on the cloud has also affected us on how we do things, when we do things, if we going to shut down the equipment we only allow for two hours, if we're gonna do maintenance it's gonna be longer than 2 hours, it has to be done on the weekend."
PA3: "So we're looking at performance, security, sociability between products, our product has to work with other products, for example; so product A, the product we build here, needs to socialize with product B and product C (B and C done by others, US team). And most of the time it's about data. The way we do it is, whoever releases, has to make sure that it's socialized with previous version, whenever we release it, have to make sure we run back to other products that interact with us, make sure APIs is talking to each other, and make sure the correct data is being sent across."
PA3: "The one thing that we have noticed with them, because in the past, everybody was doing feature testing and this is the second project that we're moving to the new model where we ask them to look at the big picture. Because the development teams themselves are making sure the features they're releasing have good quality. But they [Bangalore, India) still keep going back to say we want to know every single detail about the feature.and we keep enforce it to them, forget about the feature, you can assume that embedded QA has done 100% of that, you make sure we don't break any integration points. That how the Bungalow team is acting at the moment, is an independent and specialized team."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA3: “We also have an independent team, which is the Bungalow team, they don’t look so much at the features that we’re building, but they’re looking at the overall process. For them, a big environment, so everything can just be in one box. we still have a plan, that each team needs to deliver good quality and zero escaped defects, then we got the plan for independent team, to ensure migration, performance, sociability.”
PA3: “We do migration as well, just make sure when we go to the cloud, and migrates properly. In that process, we tend to interact directly with operations USA based], so we give them a copy of the latest backup from production, we give them the migrations as scripts, they run that through and that determine whether we can go and migrate the product during the week or whether it’s gonna be done over the weekend.”
PA3: “Yes, I think the big eye opener for us is that the operation team is completely independent body based in US, so interacting with them, sometimes we know what the problem is, will be quite easy for us to say just get us the access to machine and I’ll fix it on the spot.But with operation, it doesn’t work that way. There’s an issue, there’s the ticket, and you provide a solution, so it works by raising the ticket.”
new: Lack of architecture-organization alignment
PA1: “the ability to deliver business value.”
PA2: “challenge and lacking the expertise of integrating all the different products and offerings under one company product. Hired a VP of architecture (VIA), someone who is officially in-charge of the product architecture for the entire organization.”
PA2: “challenge is to provide clear information, guidance and support for each delivery team on enterprise architecture.”
PA2: “challenges in incorporating new and emerging technologies, that number is growing, I think there’s a lot more custom adapters out there, there’s one or two items that had to do with other technology, some new tools, some API, some metadata they got, that we need to consume and fit into our model.”
PA2: “One application one tenant, we call that single tenant database, we considered it, and this was before we moved to the cloud, maybe 5-6 years ago. It was a massive decision, in fact in the end, the engineering decision, we had a consensus and that was, you need both, but the less risky one, the one that’s actually going to get us quickly to being able to do multi-tenant would be separate instances, but in this world where we might have 5000 customers, we don’t want 5000 small database to manage. We would need the multiple tenancy in one database. We wanted to stage that, we want to do the single instance one first, then we’ll do multi-tenant database later. It was completely reversed by management, upper senior executives.”
PA2: “security in multi-tenancy, from data bleed, we sort of solved this within the SQL server itself. We have these multi-tenant tables, that has all the tenant’s data mapped and merged together, there’s always a tenant ID, there’s always an index on that particular tenant. Obviously that’s now enough, because you have to remember to filter everything by the tenant. So what we’ve done now, we basically got the rule, to say, nobody except exceptional circumstances, is allowed to access those tables. What we do instead, we create these views, that basically 99% of our current users, and that will automatically apply, essentially that filter for the tenant.”
PA2: “the issue of geographical locations not having a shared understanding on enterprise architecture.”

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA2: “We call that single tenant database, we considered it, and this was before we moved to the cloud, maybe 5-6 years ago. It was a massive decision, in fact in the end, the engineering decision, we had a consensus and that was, you need both, but the less risky one, the one that’s actually going to get us quickly to being able to do multi-tenant would be separate instances, but in this world where we might have 5000 customers, we don’t want 5000 small database to manage. We would need the multiple tenancy in one database. We wanted to stage that, we want to do the single instance one first, then we’ll do multi-tenant database later. It was completely reversed by management, upper senior executives.”
PA2: “we don’t know how that piece talk to our product and so we need to learn about how they hang together.”
PA2: “We have a component in our software that gathers data from a lot of different data sources, about 20 adapters to various technologies that generally include inventory gathering tools and that kind of thing. And that number is growing, I think there’s a lot more custom adapters out there, so that’s a continuous from every release, there’s one or two items that had to do with other technology, some new tools, some API, some metadata they got, that we need to consume and fit into our model.”
PA2: “Yes, obviously with architecture, sometimes you have to change as you learn things, especially when you’re dealing with unknown technologies. You have to even change your architecture at times to support those changes.”
PA2: “Yes, we have on the cloud for example, we have about 5 sort of databases, each hosting a number of tenants, at least 2 different geographies, US and Europe. I think our biggest database has got like 50 tenants in it, and the other are starting to fill up as well, and we’ve reached the limit before, and we just simply start up another server. It is not a simple matter of one server, you have to, it’s all clustered machines, you have to through 3-4 different servers you have to bring up to every new database.”
PA2: “Yes, we have on the cloud for example, we have about 5 sort of databases, each hosting a number of tenants, at least 2 different geographies, US and Europe. I think our biggest database has got like 50 tenants in it, and the other are starting to fill up as well, and we’ve reached the limit before, and we just simply start up another server. It is not a simple matter of one server, you have to, it’s all clustered machines, you have to through 3-4 different servers you have to bring up to every new database.”
PA3: “We still have designs; the AO is responsible for the overall. The whole team get involved in that, working in a feature, and AO work with the team to design it.”
PA8: “Yes, like I said, performance is a constant architectural challenge and we’ve really struggled with it in the past. Lot of our things in the past were being built without really thinking about it till the end. And we ran into huge problem that way, so we’ve tried to reverse that, we’re tried to push that into the architecture right at the start.”
PA8: “Yes, like I said, performance is a constant architectural challenge and we’ve really struggled with it in the past. Lot of our things in the past were being built without really thinking about it till the end. And we ran into huge problem that way, so we’ve tried to reverse that, we’re tried to push that into the architecture right at the start.”
new: Lack of face-to-face interaction inhibits knowledge sharing

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA3: "From there, part of the new process is that engineers actually get to estimate these things, although we have our program management and leadership roles estimate things, their estimate isn't really concrete until the engineers and the team members actually go and estimate that work, because they are the ones who's got to be doing them, if they think it's gonna be double the amount of time, probably will be double the amount of time."
PA3: "It worked well when they were related squads, if they are on the same team, they can communicate with each other quite well and help each other out. The thing that I liked about it, is the ownership, you know the go-to people for this piece of code. That person, they're expert in that sort of thing."
PA3: "They will typically have stories points, usually it's put by the PO, so not necessarily by the people who actually do the work."
PA7: "DevOps is another one, it needs to be planned and executed proactively, otherwise you'll run into troubles, so that's the way we deploy the product and the way we communicate with operations about how they're gonna do it and we meet with them, and we have an installation guide they run through, one for UAT, one for production, and they check all boxes to make sure that they have done. everything."
PA7: "For instance, now, we've got an independent test team member who has join our team in inception, but that's not always the case."
new: Lack of process alignment
PA2: "So definitely more focusing on team collaboration. Self-organizing. Different team will do things differently within reasons, some teams will have great big whiteboard on the wall, post-it notes, while another team is completely in Jira, not doing anything on whiteboard."
PA2: "the AO who's writing up the stories, but a lot of the stories were very technical. I will say it should be the PO, we should be coming into the project with these already written up and well defined and it's up to team to maybe refine them more and ask question, but it should already be there."
PA3: "It was a bit of caution meeting to assign story points, especially since story points are different between teams. In the last project, we started to compare story points between teams."
PA3: "Pairing 2 people on 1 keyboard, working together, that does happen. The kind of pairing I see more often is 2 or more people working on the same thing, but each of them work at their own desk, they'll get up and walk over and walk back, lots of movements, definitely making use of all the resources."
PA3: "Pair programming, maybe guidelines, I think each team need to figure out how they work best for themselves. Maybe the PO say maybe should try pairing more, and they'll try it, and if it doesn't work, and in the future, someone else said that again, they will say it doesn't for them."
PA7: "Again, it depends on the team, the team I was in would do that, we had AO who get up and draw diagrams, write out stories and acceptance criteria on the board."
PA7: "Iteration planning, it's as long as it takes, it depends, different teams take different amount of times, in the team that I used to be in, it took maybe an afternoon, other teams, that's taken a day or longer."
PA7: "That's what they should be, in my old team, we did pretty well, stories were put into a form of acceptance criteria with extra context, and we wouldn't even have thought of doing it differently. My new team, I'm trying to get them to do it, it's a bit hard, but we'll get it there."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA7: "User stories, the way we did it in my old team, usually AO write it on the board, we take a photo and that's your index card."
new: Lack of tool/infrastructure alignment
PA2: "We got multiple database servers in EU and US, and even just allowing access to engineer into the system is not gonna happen, only a select group of people. That doesn't mean the people that get to deal with the task and issue, so if I get a support ticket from customer in the cloud, I need to go to one of them and ask for access into the system and database, so that I can start working on it. Until I can do that, it's difficult, it's impossible to make progress."
PA3: "Greylog was quite painful, back when we were new to cloud, we had one of the old version of Greylog, the person who looked it up in the operation has quit, so no one knew how it worked, and it's my job to go in there every day and see what's going on the cloud, and it's a manual process, I couldn't get any support to try to automate it. So, I don't know about learning curve, but it was definitely painful. Things like New Relic, the people who need that information, for them it's not that bad learning curve, because they know what they want to see and they go in there, and it's there, for people who don't use it day to day, it's still a bit of a mystery."
PA3: "No, we still have to sit and wait, and I think they're getting slower and slower. Every now and then there'll be something, a few years ago, one of our staff engineer moved us to MS Build, I think that speed things up quite a bit, made it a lot nicer."
PA3: "one resource that's running low is meeting rooms, because we got so many teams, 4 teams, management teams, sales people, everyone is using them. That's an important resource and proximity is important. Sometimes certain meeting rooms have A/V equipment, when we have people who want to use that room or checking in with non-co-located like in India, Bungalow, but the room is booked by another team, it can cause all sorts of issues."
PA3: "TDD–we had what we called quality initiative, where we focused very heavily on improving our testing infrastructure and writing test for areas that didn't have tests, since then it's been much more focus making sure we do unit testing. At that point, I think the law came down from above, that every time you fix a bug, you must write a test for it."
PA3: "We have a lot of tests, and different levels, there was an effort to split out the unit test, but I think that didn't really happen as well as people were hoping, beyond that level we also have migration test, takes a long, also have QA system test, they take a long time. Then there's QA performance test, I'm not even sure when and how they run."
PA3: "We're using visual studio to build our stuff, we're using new stuff for kind of like monitoring and logging with Greylog, New Relic for performance, and few things like that we never used before."
PA3: "We used to have schedule task that will run the builds, we move over to Jenkins system, where if something is changed, kick off test, it's a lot better now. But still a long time before you know what is broken or something."
PA6: "we run our own monitoring tools and we collect the data, so we got one component called the 'importer', what it does is imports all sorts of information from database into our system and that probably the biggest one that gives us the biggest headache. We monitor that daily and we get emails whenever one of those fails, so we can start and identify the change we can make. But the monitoring we tend to create it and we use it rather than the operations. And that's how we feed those information back to development."

Continued on next page

Table 5 (continued): Case A issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PA6: "Yes, we haven't achieved 100% automation, a lot of people think it's the solution to everything. But you build a component, even though we ask you to provide 100% automation for cloud and on-prem, there's always a particular failure So there's component of manual testing and also exploratory testing that we do. Automation has played a bigger role for us, we reduced the release cycle from 6 weeks to 2 weeks, without automation to give us that nightly check ability, it's very difficult, and our product is very complex, just setting up the environment and do manual testing is weeks of effort."
new: Unstable country/regional political/economic environment
(not observed in case)
Organization undergoing restructuring during the project
(not observed in case)
Resources shifted from the project due to changes in organizational priorities
(not observed in case)
Unstable organizational environment
(not observed in case)

4.2 Risks Seen in Case B

Table 6: Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Quadrant 1: Customer Mandate
Conflict between users
(not observed in case)
Lack of cooperation from users
PB1: "I think the biggest challenge when it comes to the client and things we need to know from them and the follow up... we need to do... a little bit more internally you know."
PB11: "... now, in an Agile world there is no way that I could tell them when they are going to get done until the estimate is there, until we start a sprint planning... you can't just say, 'we are doing Agile, so, you got to wait for our next planning...'
Lack of top management support for the project
Observation: Upper management occasionally overules decisions
Lack of user participation
(not observed in case)
Lack or loss of organizational commitment to the project
(not observed in case)
Users not committed to the project
(not observed in case)
Users resistant to change

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
(not observed in case)
Users with negative attitudes toward the project
(not observed in case)
Quadrant 2: Scope and Requirements
Conflicting system requirements
(not observed in case)
Observation: Large customer teams push for their fixes
PB3: "... [T]he integration required ... [is] not only the standard [flagship product] patient work journey. But it's the patient journey customized for [customer] in which there is... insurance and membership and insurance plans and so it is a lot more complex than the standard [flagship product]. And that process was painful because of... the geographical distance."
PB14: "They [global customer] want a lot of stuff changed and then the main problem they have [is] localization. Insurance is a big part in [the] [specific region] and it's very difficult there. I think it's the most difficult and challenging part in our software."
Continually changing project scope/objectives
PB3: "... [S]ometimes customers are asking for [us] to sign-off and provide estimates and a statement of work on something that we still don't have a full understanding and full specifications... There could be high risk of... adding scope and having [to spend] more budget than originally forecasted... "
PB14: "They [global customer] want a lot of stuff changed and then the main problem they have [is] localization. Insurance is a big part in [the] [specific region] and it's very difficult there. I think it's the most difficult and challenging part in our software."
Continually changing system requirements
(not observed in case)
PB16: "I need to get approval from too many people. So far I have mentioned [role X] and [role Y] before putting into the backlog. That's not all of it. Some things have to go to [role Z] and especially any primary screen of [flagship product]. So, I need to get approval from too many people–one person says 'okay, good; check with X' or, 'no... take it back.' It is too difficult to satisfy that many people."
PB14: "They [global customer] want a lot of stuff changed and then the main problem they have [is] localization. Insurance is a big part in [the] [specific region] and it's very difficult there. I think it's the most difficult and challenging part in our software."
Difficulty in defining the inputs and outputs of the system
(not observed in case)
Ill-defined project goals
PB16: "So, in that respect I find it difficult. We are kind of defining stuff as we go. So, I haven't got strong reference point for why we are doing [the work on the project]."
Incorrect system requirements
(not observed in case)
System requirements not adequately identified

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PB4: "Like I remember a few weeks ago we were with a customer from [region] that asked for a particular improvement in a particular area, and the whole issue had gotten completely, you know, misunderstood by development and had been sitting in the backlog for a very long time, and [we] would set up a call with the customer to try and clear up exactly what they want. Because a lot of the time there is this 'Chinese whispers' thing going on with... people misunderstanding things, recording the wrong request and so on."
PB16: "So, in that respect I find it difficult. We are kind of defining stuff as we go. So, I haven't got strong reference point for why we are doing [the work on the project]."
PB11: "... the client would want to know a plan–they want to have a plan for everything that we [are] going to do. Let's say, there are five change requests, for example, that's in-progress as in, we are still talking to them about what the requirement is. So, there is no estimate on these. They want to know exactly when they [are] all going to get done."
PB10: "... it turned out it was a very difficult thing to do to write a technical specification when you are not the actual developer that's going to be developing that piece of material."
Unclear system requirements
PB16: "I need to get approval from too many people. So far I have mentioned [role X] and [role Y] before putting into the backlog. That's not all of it. Some things have to go to [role Z] and especially any primary screen of [flagship product]. So, I need to get approval from too many people–one person says 'okay, good; check with X' or, 'no... take it back.' It is too difficult to satisfy that many people."
PB16: "So, in that respect I find it difficult. We are kind of defining stuff as we go. So, I haven't got strong reference point for why we are doing [the work on the project]."
PB16: "So, in that respect I find it difficult. We are kind of defining stuff as we go. So, I haven't got strong reference point for why we are doing [the work on the project]."
PB16: "We kind of define stuff as we go. So I haven't got a strong reference point for why we are doing [the work in the project]
Undefined project success criteria
PB16: "So, in that respect I find it difficult. We are kind of defining stuff as we go. So, I haven't got strong reference point for why we are doing [the work on the project]."
Users lack understanding of system capabilities and limitations
(not observed in case)
Quadrant 3: Execution
Development team unfamiliar with selected development tools
(not observed in case)
Frequent conflicts among development team members
PB10: "... [W]e also had lot of conflicting sort of decisions within. We had [porfolio role] who is the [team role], and a [Sr Developer] who is the Technical... so that's been quite difficult as well."
Frequent turnover within the project team
(not observed in case)
High level of technical complexity
Observation: [a given agile team role] raises more issues than are solved

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Highly complex task being automated
(not observed in case)
Immature technology
(not observed in case)
Inadequate estimation of project budget
Observation: Estimates vary widely from reality, usually too low
Inadequate estimation of project schedule
PB9: "We kind of overcommitted the first time. We overcommitted to a lot of stuff, work to be done within the sprint which I think was too much. It is a result of bad or wrong estimation. So, whatever we committed to do in two weeks in reality the job took four weeks."
PB13: "From my own point of view if [X] says a ticket takes five days and as a developer if I think it will take 10 days then I probably won't say 10 days because other people may think I am taking more time than I should. The reality is everyone takes a longer than the initial estimation."
PB10: "For the first few sprints we completely overcommitted to a lot of stuff which we just couldn't deliver. So, we are trying to fit in with the velocity that is based on the size of the team."
Inadequate estimation of required resources
Observation: Shortage of testers
PB15: "I was working with the [A] team and there is also [B] project as well. So, it's like 50% of my time designated for project [B] and other 50% for other projects such as [A] and also [C]. ... I don't have the full picture of everything that's going on. Again, when you've got 50% time on this project then you do tend to miss of bit of both."
PB15: "When you got 50% time on this project then you do tend to miss a bit of both"
PB4: "... I see the bottleneck and the delay that the whole QA process is causing. Don't get me wrong we have brilliant QA people but ... too much going on to satisfy all of the demands from all the different projects etc. ... [I]f we had automated QA it would release code a lot more often and you know basically keep the customer a lot more up to date in their versions."
PB12: "We still try to complete the QA within the sprint. Because this is one of the big issues we are having certainly earlier of the sprint. Most of the tickets delivered to the QA on second or last day of the sprint then QA doesn't have enough time to close them. ... that was the problem we are having."
PB1: "Currently I am working on [X] project and then [Y] project. Those two are client facing. Then I also have the [Z], what we call the '[Z for country A]' product, which is [flagship product] integrated into [Application]. So that's... another one that I'm taking on as well."
PB1: "I think in that sense, normally, I noticed that the internal projects will always kind of suffer, sometimes it's fine because the team is just there on autopilot kind of thing and they are moving along, but then I kind of sometimes also feel that things would get left behind just because no one is there to keep asking about it or just following up type of thing."
PB1: "It's very challenging. But, what happens is the one that is not client facing, is always being treated like a neglected child... There'll be days like I only spend [the] minimum... just on our standup for example, and then if there [are] any blockers that they need my help [on], I'll do that. But that will be a bare minimum."

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PB1: “The two that are client facing, they kind of take priority in terms of time, and then I normally do [. . .] between those two almost 70%, and then the little bit that’s left over I’ll just do our [maintenance tasks].”
PB9: “Lack of coordination with QA, QA side was not really had the time to finish the task within the sprint–it poses carry over. Which shouldn’t be the case where proper estimation including the QA time, it should stick within the sprint. It should not go over or go beyond.”
PB9: “On day eight QA receive multiple tickets from multiple developers and there is only one QA.”
PB13: “. . . as developers are uncertain which machine to use and which is available. It’s a disruption for us on a daily basis. We have five active versions in build machine. So if someone is accessing the build machine right now then I won’t be able to know when they will be finished. I don’t know when I will get the access.”
Inadequately trained development team members
PB10: “In this particular project [motivation is] neither low nor high because of the learning curve. And, it’s been very difficult to wrap my head around the role itself. . . . [I]t’s been quite demoralizing actually when you don’t fully understand your job properly.”
PB10: “. . . myself and [X], even though we did get a bit of training for the [agile team role] stuff, I don’t feel like the team had enough training on what our role was. So, I think our role wasn’t done to the best of its ability.”
Ineffective communication
PB9: “A classic example in [large customer project] would be having the file from the [project A]. If we don’t have file then there is no way we can test it. So, these are the things that could cause delays or some challenges within the sprint.”
PB14: “Possibly things would be late if you do not have face-to-face communication we can interpret the things in different way.”
Ineffective project manager
(not observed in case)
Inexperienced project manager
(not observed in case)
Inexperienced team members
(not observed in case)
Lack of an effective project management methodology
PB1: “. . . you got two product owners and you’ve got each product owner saying ‘This is a top priority, work on this”’
PB14: “. . . it looks like we have waterfall. But, we don’t have waterfall, so we are trying to be more Agile.
Lack of commitment to the project among development team members
Observation: Team members report neutral motivation
Lack of people skills in project leadership
(not observed in case)

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
Large number of links to other systems required
Observation: Changes made by customer teams break features
PB2: "We try really hard to get, you know, to a common place. It is very difficult with lots of different paid development and, you know, and chains having the same product as the independents. That's difficult as well. So, you know, we're really up against it. But we really want to do it."
Negative attitudes by development team
Observation: "We don't trust our [role in agile team]"
new: Ineffective collaboration
Observation: Changes made by customer teams break features
PB4: "Like I remember a few weeks ago we were with a customer from [region] that asked for a particular improvement in a particular area, and the whole issue had gotten completely, you know, misunderstood by development and had been sitting in the backlog for a very long time, and [we] would set up a call with the customer to try and clear up exactly what they want. Because a lot of the time there is this 'Chinese whispers' thing going on with... people misunderstanding things, recording the wrong request and so on."
PB1: "I think the biggest challenge when it comes to the client and things we need to know from them and the follow up... we need to do... a little bit more internally you know."
PB1: "... [A]ll teams were doing their own thing in term of releases and it's just very independent of one anotherldots"
PB9: "We kind of overcommitted the first time. We overcommitted to a lot of stuff, work to be done within the sprint which I think was too much. It is a result of bad or wrong estimation. So, whatever we committed to do in two weeks in reality the job took four weeks."
PB11: "... now, in an Agile world there is no way that I could tell them when they are going to get done until the estimate is there, until we start a sprint planning... you can't just say, 'we are doing Agile, so, you got to wait for our next planning...'
PB10: "... [W]e also had lot of conflicting sort of decisions within. We had [porfolio role] who is the [team role], and a [Sr Developer] who is the Technical... so that's been quite difficult as well."
new: Ineffective coordination
PB15: "I was working with the [A] team and there is also [B] project as well. So, it's like 50% of my time designated for project [B] and other 50% for other projects such as [A] and also [C]. ... I don't have the full picture of everything that's going on. Again, when you've got 50% time on this project then you do tend to miss of bit of both."
PB4: "... I see the bottleneck and the delay that the whole QA process is causing. Don't get me wrong we have brilliant QA people but ... too much going on to satisfy all of the demands from all the different projects etc. ... [I]f we had automated QA it would release code a lot more often and you know basically keep the customer a lot more up to date in their versions."
PB12: "We still try to complete the QA within the sprint. Because this is one of the big issues we are having certainly earlier of the sprint. Most of the tickets delivered to the QA on second or last day of the sprint then QA doesn't have enough time to close them. ... that was the problem we are having."
PB16: "Due to time zone barriers things could be blocked. Sometimes I'd get the answer at 9pm."
PB1: "... [A]ll teams were doing their own thing in term of releases and it's just very independent of one anotherldots"

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PB1: "... you got two product owners and you've got each product owner saying 'This is a top priority, work on this"'
PB1: "Major obstacle would be with the dedicated team and then if you have people cross country."
PB9: "A classic example in [large customer project] would be having the file from the [project A]. If we don't have file then there is no way we can test it. So, these are the things that could cause delays or some challenges within the sprint."
PB9: "Lack of coordination with QA, QA side was not really had the time to finish the task within the sprint–it poses carry over. Which shouldn't be the case where proper estimation including the QA time, it should stick within the sprint. It should not go over or go beyond."
PB9: "The Build machine does have an effect in some cases. For example, if I send something to QA and QA will come back and [be] saying, 'What would be [the] starting point for the test?' In some cases, I don't even know at what point the QA is testing it. So, I have to setup the testing environment and I have to make sure that the setting is turned on [in] that environment for her to test it."
PB7: "Running global, multinational and even just dispersed across the countries, teams, all of those things have their own challenges; coordination, multiple time zones, culture, normal challenges that you would imagine from having dispersed teams. If you were to turn the question on its head you consider how much easier it is if everyone is just sitting in one room to work on a project as opposed to separating them. So there is a big challenge with that."
PB13: "... as developers are uncertain which machine to use and which is available. It's a disruption for us on a daily basis. We have five active versions in build machine. So if someone is accessing the build machine right now then I won't be able to know when they will be finished. I don't know when I will get the access."
PB13: "... it happens regularly in every week – that someone forgets to unlock the unit. So, when that happens–previously all developers were in [local office] and that shouldn't be a problem–now we have a problem."
PB13: "When we do our coding we have to lock our unit that we are working on. When we finish the job we commit the code then release the lock, so someone else can check that unit about what changes. But, sometimes – well it happens regularly in every week – that someone forgets to unlock the unit."
PB11: "The challenge is if the customer (in a different continent) raised a question obviously the guy in [local team] won't be able to fix it immediately. So, now we are missing one full day for them to look at them again."
PB2: "We try really hard to get, you know, to a common place. It is very difficult with lots of different paid development and, you know, and chains having the same product as the independents. That's difficult as well. So, you know, we're really up against it. But we really want to do it."
new: Lack of trust
Observation: Upper management occasionally overules decisions
Observation: "We don't trust our [role in agile team]"
PB16: "I need to get approval from too many people. So far I have mentioned [role X] and [role Y] before putting into the backlog. That's not all of it. Some things have to go to [role Z] and especially any primary screen of [flagship product]. So, I need to get approval from too many people–one person says 'okay, good; check with X' or, 'no... take it back.' It is too difficult to satisfy that many people."

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
PB16: "Yeah, when I say rework its not re-edit. It's kind of starting from scratch. So, that is one thing I need to change because I don't think it follows agile to require approval from so many different points."
PB13: "From my own point of view if [X] says a ticket takes five days and as a developer if I think it will take 10 days then I probably won't say 10 days because other people may think I am taking more time than I should. The reality is everyone takes a longer than the initial estimation."
One of the largest projects attempted by the organization
(not observed in case)
Poor project planning
PB1: "... [A]ll teams were doing their own thing in term of releases and it's just very independent of one anotherldots"
PB9: "You could expect there will be a bottleneck during that time, if it was scheduled properly, job and tasks are completed. In reality, what happens, all developers finish their tasks around the same time then it's required a lot of responsibilities and time for QA to finish."
PB2: "It's difficult because we are still in our transition phase in putting all of this inn place. It's kind of difficult to have with so many different teams."
PB2: "We tried really hard. It is not in all impossible but we did try and we have succeeded to some extent. And I'd definitely think that if everybody sprints, just the opinion if everybody use sprint team, I think it would be easier to do."
Project affects a large number of user departments or units
(not observed in case)
Project involves the use of new technology
(not observed in case)
Project involves use of technology that has not been used in prior projects
Observation: "We decided not to do automated testing because this is a legacy product"
Project milestones not clearly defined
PB1: "... [A]ll teams were doing their own thing in term of releases and it's just very independent of one anotherldots"
PB9: "I think what would help is proper scheduling of the completed tasks: 'definition of done.' If I am done... if we have a good idea to finish something by Tuesday then at least QA will be doing something that day within the sprint. Everything now going to be three days prior to end of the sprint."
PB2: "It's difficult because we are still in our transition phase in putting all of this inn place. It's kind of difficult to have with so many different teams."
PB2: "We tried really hard. It is not in all impossible but we did try and we have succeeded to some extent. And I'd definitely think that if everybody sprints, just the opinion if everybody use sprint team, I think it would be easier to do."
Project progress not monitored closely enough
(not observed in case)
Team members lack specialized skills required by the project

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
(not observed in case)
PB10: “In this particular project [motivation is] neither low nor high because of the learning curve. And, it’s been very difficult to wrap my head around the role itself. . . . [I]t’s been quite demoralizing actually when you don’t fully understand your job properly.”
Team members not familiar with the task(s) being automated
(not observed in case)
Quadrant 4: Environment
Change in organizational management during the project
(not observed in case)
Corporate politics with negative effect on project
(not observed in case)
Dependency on outside suppliers
PB3: “Risk identification is also very crucial. And especially when we are dealing with third parties [it] is kind of challenging.”
PB8: “Well, we categorized our epics into different sections and modules based on the multiple different moving parts of the product such as interfaces, contact lens interface, third party company. . . ”
Many external suppliers involved in the development project
(not observed in case)
new: Country-specific regulations
PB3: “. . . [T]he integration required . . . [is] not only the standard [flagship product] patient work journey. But it’s the patient journey customized for [customer] in which there is. . . insurance and membership and insurance plans and so it is a lot more complex than the standard [flagship product]. And that process was painful because of. . . the geographical distance.”
PB6: “ldots [I]f I talk about [the] data team we have GDPR regulations coming. . . which means we need to start encrypting databases. That’s a regulation or legislative regulation coming in [which] means we must do that by a period of time.”
PB2: “. . . [W]hen we try to observe HIPAA, and because, although that’s only in the US at the moment, we might as well just observe it everywhere.”
PB5: “It is super specific in France, the vertical business is not driven by medical or exam it is mainly driven by retail because it is purely retail and there are many regulations, constraints and rules. So one of the projects we have to manage especially for January [for] example we have new regulations to integrate. . . for the customer.”
PB5: “You have the regulations, fiscal, the way you calculate the TVA, the way you do invoicing, but also the core business of optic is completely different. . . . [I]n France you go to the doctor, to the opthamologist, and he will deliver the prescription and after that you go to the store, to the optician, to buy your pair of spectacles, your equipment. . . In France the optician is not allowed to do some medical exam and to do the prescription. . . ”
PB10: “. . . [Y]ou have to consider what the user sees on the screen and, you know, if other people might be able to see it. And you have to consider how, for example, the text reminders we send, you know, they’ve got to be secure, nobody can hack-in and steal them.”

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
new: Delays caused by global distance
(not observed in case)
PB3: "At times there are inevitable delays due to the geographical distribution of the team. . . "
PB16: "Due to time zone barriers things could be blocked. Sometimes I'd get the answer at 9pm."
PB1: "Major obstacle would be with the dedicated team and then if you have people cross country."
PB7: "Running global, multinational and even just dispersed across the countries, teams, all of those things have their own challenges; coordination, multiple time zones, culture, normal challenges that you would imagine from having dispersed teams. If you were to turn the question on its head you consider how much easier it is if everyone is just sitting in one room to work on a project as opposed to separating them. So there is a big challenge with that."
PB13: ". . . it happens regularly in every week – that someone forgets to unlock the unit. So, when that happens–previously all developers were in [local office] and that shouldn't be a problem–now we have a problem."
PB13: "When we do our coding we have to lock our unit that we are working on. When we finish the job we commit the code then release the lock, so someone else can check that unit about what changes. But, sometimes – well it happens regularly in every week – that someone forgets to unlock the unit."
PB11: "The challenge is if the customer (in a different continent) raised a question obviously the guy in [local team] won't be able to fix it immediately. So, now we are missing one full day for them to look at them again."
PB10: "It doesn't help, the fact that the customer is actually [in a different continent]. The difficulty there is that, when I am here on-site I only get a couple of hours in the morning to deal with [local] stuff. So if I don't get the things that I need from them within those two hours, I'm pretty much isolated for the day."
PB10: "So, unfortunately, when I am here on-site [X] has own number of epics, which I can't answer when I am here. And, same for him–when he is here he can't answer the epics I worked on."
PB10: "The difficulty is that, when I [agile team role] am here on-site I only get couple of hours in the morning to deal with [local] stuff. If I do not get the things that I need from them even though these two hours I am pretty much isolated for the day. Even though I have Technical Lead here in [city] he is very heavily involved with other things."
new: Lack of architecture-organization alignment
PB15: "When you got 50% time on this project then you do tend to miss a bit of both"
PB3: "At times there are inevitable delays due to the geographical distribution of the team. . . "
PB13: ". . . it happens regularly in every week – that someone forgets to unlock the unit. So, when that happens–previously all developers were in [local office] and that shouldn't be a problem–now we have a problem."
PB10: "So, unfortunately, when I am here on-site [X] has own number of epics, which I can't answer when I am here. And, same for him–when he is here he can't answer the epics I worked on."
PB10: "The difficulty is that, when I [agile team role] am here on-site I only get couple of hours in the morning to deal with [local] stuff. If I do not get the things that I need from them even though these two hours I am pretty much isolated for the day. Even though I have Technical Lead here in [city] he is very heavily involved with other things."

Continued on next page

Table 6 (continued): Case B issues mapped to GSD Risk Catalog risks

GSD Risk Catalog risk/Issues
new: Lack of face-to-face interaction inhibits knowledge sharing
PB15: "I was working with the [A] team and there is also [B] project as well. So, it's like 50% of my time designated for project [B] and other 50% for other projects such as [A] and also [C]. . . . I don't have the full picture of everything that's going on. Again, when you've got 50% time on this project then you do tend to miss of bit of both."
PB15: "When you got 50% time on this project then you do tend to miss a bit of both"
PB14: "Possibly things would be late if you do not have face-to-face communication we can interpret the things in different way."
PB7: "Running global, multinational and even just dispersed across the countries, teams, all of those things have their own challenges; coordination, multiple time zones, culture, normal challenges that you would imagine from having dispersed teams. If you were to turn the question on its head you consider how much easier it is if everyone is just sitting in one room to work on a project as opposed to separating them. So there is a big challenge with that."
new: Lack of process alignment
PB2: "We tried really hard. It is not in all impossible but we did try and we have succeeded to some extent. And I'd definitely think that if everybody sprints, just the opinion if everybody use sprint team, I think it would be easier to do."
new: Lack of tool/infrastructure alignment
(not observed in case)
new: Unstable country/regional political/economic environment
(not observed in case)
Organization undergoing restructuring during the project
PB10: "So, we worked in waterfall fashion in the past and I think this is difficult for people to move from the waterfall way to the Scrum way."
Resources shifted from the project due to changes in organizational priorities
(not observed in case)
Observation: Large customer teams push for their fixes
Observation: PO responsibilities split between support and PO
PB13: ". . . as developers are uncertain which machine to use and which is available. It's a disruption for us on a daily basis. We have five active versions in build machine. So if someone is accessing the build machine right now then I won't be able to know when they will be finished. I don't know when I will get the access."
Unstable organizational environment
(not observed in case)

References

[1] S. Ambler and M. Lines. *Disciplined agile delivery: an introduction.* IBM Software, Somers, NY, 2011.

[2] Sarah Beecham, Tony Clear, Ramesh Lal, and John Noll. Do scaling agile frameworks address

global software development risks? an empirical analysis. *Journal of Systems and Software (JSS) (in review)*, 2020.

[3] Sarah Beecham, John Noll, and Mohammad Abdur Razzak. Lean global project interview protocol. Technical Report TR_2017_02, Lero, the Irish Software Research Centre, 2017. http://www.lero.ie/sites/default/files/Lero_TR_2017_02_Beecham_Noll_Razzak-Lean%20Global%20Project%20Interview%20Protocol.pdf.

[4] R. Lal and T. Clear. Scaling agile at the program level in an australian software vendor environment: A case study. In *Australasian Conference on Information Systems*, pages 1–12, 2017.

[5] R. Lal and T. Clear. Enhancing product and service capability through scaling agility in a global software vendor environment. In Roberto Espinoza and Darja Šmite, editors, *Proceedings 2018 IEEE 13th International Conference on Global Software Engineering*, pages 59–68, Los Alamitos, California, 2018. IEEE.

[6] D. Leffingwell, A. Yakyma, R. Knaster, D. Jemilo, and I. Oren. *SAFe 4.0 Reference Guide: Scaled Agile Framework for Lean Software and Systems Engineering.* Addison-Wesley Professional, 2016.

[7] Mary L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

[8] John Noll, Sarah Beecham, Ita Richardson, and Clodagh NicCanna. A global teaming model for global software development governance: A case study. In *International Conference on Global Software Engineering*, pages 179–188, Irvine, California, USA, August 2016. IEEE.

[9] John Stouby Persson, Lars Mathiassen, Jesper Boeg, Thomas Stenskrog Madsen, and Flemming Steinson. Managing risks in distributed software projects: an integrative framework. *IEEE Transactions on engineering management*, 56(3):508–532, 2009.

[10] Mohammad Abdur Razzak, John Noll, Ita Richardson, Clodagh Nic Canna, and Sarah Beecham. Transition from plan driven to SAFe®: Periodic team self-assessment. In *International Conference on Product-Focused Software Process Improvement (PROFES '17)*, pages 573–585. Springer, 2017.

[11] J.M. Verner, O.P. Brereton, B.A. Kitchenham, M. Turner, and M. Niazi. Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, 56(1):54–78, 2014.

[12] L. Wallace and M. Keil. Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4):68–73, 2004.

Appendix A DAD/CASE A: Study and Interview Protocol

Authors: Ramesh Lal & Tony Clear, AUT (study wholly undertaken by AUT researchers). Published in this Technical Report as supplementary information for the Lero/AUT collaboration on Risk in GSD project.

A.1 Main method used to collect data

In-depth interview,

- Based on the DAD framework identify key constructs to develop interview guide and write in-depth interview questions.
- Questions asked in a systematic way.
- Identify and request appropriate persons in various DAD roles, to gather extensive information.
- Have full cooperation from the participants. Participation must be voluntary

A.2 Roles requested and agreed for interviews

Director of software engineering, engineering manager, program manager, team lead, architect owner, quality assurance manager, product owner, software engineers. These were engineers (post-graduate, senior, and principal levels) reflecting their experiences of software development.

A.3 Interview Instruments

The following were the interview instruments or key themes based on which individual interview questions were formulated, DAD adoption, DAD phases, DAD method roles, Product and portfolio management, program management, and project management.

Each interview session was planned for an hour.

Interview sessions were held in the meeting rooms at the production lab of the case study participants. In total 8 sessions were held in a week in March 2017.

The requests for interviews were made through the Director of software engineering and managers of the agile teams of two participating organisations. A timetable for interviews was agreed with the director of software engineering.

Interviewees were informed of their right not to answer any particular question they were not comfortable with.

Participant anonymity was also ensured. The impact on their work schedule was minimised by having interviews early in the mornings, late afternoons or during lunch time (as convenient to the individual), and limiting the interview sessions to one hour at a time

It was also agreed that the case study report rather than individual interviews would be validated to minimize impact on their development commitments.

Agreement was made on recording of interviews and individual consent was taken before a session began. These recorded interviews were later transcribed. Recording enabled accuracy for data collection and to be more focused on the interviewee. However, important notes or emerging questions were noted down during the session.

The interview questions were knowledge questions about their DAD adoption

Prior to an interview session all questions were read and revised. Questions were also read out to colleagues to ensure their clarity. Any probing or follow-up questions resulting from the previous session were included.

Each interview was recorded, labelled and transcribed. The transcribed interviews were saved as Microsoft Word documents using interviewee title.

A.3.1 Software Engineer, Primary role–31st March 2017

1. From a software engineer's perspective what does being disciplined mean to you?
2. DAD team roles PO, TL, AO & domain support, how have these roles helped you to move fast? What Improvements have been made compared to the hybrid approach?
3. Inception phase, when you work with AO, is it a team effort?
4. 3 leadership roles in your team, what is your view on that when it comes to coordination activities in the team?
5. Issues and difficulties in your projects when you adopted DAD?
6. Definition of done (DOD), do software engineers have any input in defining DOD?
7. With the DAD approach, what are your TDD practices and has quality improved?
8. Can you identify all the Practices that you have or use in your DAD team?
9. So, is every team member responsible for project management (participate in all phase activities of your DAD approach)?
10. How do you estimate (practices) tasks (user stories)?
11. Do you plan pair programming when you assign tasks?
12. TDD and you have QA embedded, do you have to have the testing mindset now?
13. Do the system test happen in every sprint now.
14. Who defines and writes acceptance test criteria?
15. What else you have in your sprint? Any other practices?
16. What are some of your practices in the transition phase?
17. In the hardening (transition) phase, if you fix a bug, do you have to rerun all the test?
18. With DAD are there more technical skills you need now as software engineer?
19. With DAD approach now you have to think and plan about deployment as well, not just integration and delivery (development and testing) so is more technical knowledge and understanding about operations needed?
20. What about monitoring deployed product, as a software engineer do you get real time data?

A.3.2 Engineering Manager, Team Lead (TL)–30th March 2017

1. As TL are you the product development manager as well?
2. What do you feel about the DAD approach?
3. Your responsibilities as TL?
4. Did you face challenges (and what were some of the challenges) when you switched to this structure?

5. Can you identify key characteristics of your squads from your hybrid approach?
6. Which one do you prefer for project teams, large DAD team or squad structure?
7. With your DAD approach, in terms of your collaboration with QA, documentation, UX teams? How frequently are they are involved in your project?
8. Does documentation team still have and bring the usability perspective into the DAD project teams?
9. Can you identify and explain the various roles in a DAD project team–the primary and secondary roles?
10. The validation and verification of features the team produces, will it be the responsibility of the PO?
11. In terms of Team Lead (TL) role, what do you do in your team?
12. Is your daily stand-up time boxed as well?
13. What else does the team lead do besides project work i.e. in regard to program management?
14. Explain the practices of "tactical huddle"
15. Does team lead (TL) have to do a lot of collaboration with program management? Or just the POs?
16. Do you use burndown chart or any other tool for M/C??? in projects?
17. Any issues or challenges with Jira?
18. With DAD's 'T-skilled approach, does your team have any issues in certain skills or aspects?
19. The secondary roles, specialist roles–do senior engineers fill the secondary roles in a DAD team?
20. Who represents or who is your domain expert, would it be PO?
21. With your DAD approach, there would also be an Independent tester (in the secondary role) from the QA team working your DAD projects now–explain how it works?
22. What phase of your DAD project do you bring them in? For each IR?
23. In inception phase, what are the things you do to explore scope?
24. You time box your spikes?
25. Do you run spikes in iterations as well?
26. User stories? Why not use cases anymore?
27. In user stories, are you assigning points?
28. How many can you do in an iteration? Varies?
29. Iteration planning, when does it happen?
30. So, it's a team, collectively?

31. When it comes to transition phase, about to deploy, what are the activities there?
32. Who designs the UAT?
33. Any other DAD project team practices that you have?
34. Explain the DevOps practice as part of DAD approach?
35. Product backlog, before it will mostly have captured new features to be implemented, but now different compared to work item list–can explain the two concepts?
36. What are some of the rules you have for product backlog?

A.3.3 Senior Software Engineer, in Primary role–29th March 2017

1. Identify some critical new practices in your agile approach now? DAD?
2. What were the key practices in squad-based team setup?
3. When you are the squad, you are embedded in the team?
4. What was your overall size of squad teams and what is it now with DAD project teams?
5. What were some critical differences between squad and DAD teams?
6. What are your specific practices now in DAD project team?
7. What is the format for your daily stand-up meeting with DAD approach?
8. What are the benefits to the team when you have stand-up meetings in DAD approach?
9. What other critical DAD team practices do you have?
10. Explain your iteration planning practice now with DAD approach which can take a whole day?
11. Can you explain your story point allocation practice with DAD approach?
12. When you plan your Iterations, what are the rules now you must enforce?
13. Explain your estimation practice i.e. planning poker any other which you have adopted with your DAD approach?
14. Your user stories have acceptance criteria?
15. Do you use index card or what is your practice capturing a user story?
16. Who writes user stories?
17. In your DAD approach, conversations are happening freely in your team, between the developers (primary role) and PO?
18. What is your view on DAD approach?
19. When it comes to writing code? Does your team use solo, pair practices or both?
20. From your perspective which practice gives a better code quality? Pair or solo?

21. Do you still have a design session now in your inception phase?
22. With pair programming, was any training provided with driver & navigator roles?
23. With DAD, writing unit test is a key practice–what is it you feel about it?
24. Overall, how important is TDD now for continuous delivery of software in iterations?
25. You have got QA embedded in your DAD project teams, do they enforce TDD practices?
26. Is refactoring practices, part of your DOD?
27. Explain the primary role with your DAD approach in terms of the graduate, senior, staff and principal engineer roles which you have. Any new levels since DAD approach or any change in these software engineering levels?
28. How long for someone from associate to move to engineer?
29. Can you identify DAD project team challenges with technical tasks? when you shift to cloud deployment?
30. The DevOps practice, operations are in the US, so how do you collaborate with them?
31. Can you explain how your DAD project team now deals with the support ticket that is allocated?
32. The Team work area with your DAD approach now are you happy with the setup? Got whiteboards?
33. Teamwork area, what about noise? How does the team maintain (practices) a peaceful environment?
34. The infrastructure that you need IDE for cloud application development and deployment–any challenges?
35. What was the learning curve like for new tools?
36. What is the story with your built infrastructure now since iteration delivery is even more critical with DAD approach? So, you do not have to sit and wait?

A.3.4 Product Owner (PO), User Experience Specialist–28th March 2017

1. What is your role here and what you do?
2. What are the differences when you work on cloud and on-premises?
3. The UCD approach, are you following it for your DAD approach?
4. Do you have a usability lab here?
5. What are the challenges in Web UI and what specific practice do you have now with your DAD approach?
6. What were the main objectives of your role in terms of cloud approach for feature deployment that the company was embarking on?
7. Can you explain the business side of things in your role?

8. How does DAD approach enable you to come up with useful, usable, engaging product?
9. In your role as a PO, how closely are you working with PM?
10. Does DAD allow for UX point of view at product management level.
11. In cloud, do you look at the metrics?
12. Overall, with DAD approach, how does your input go into inception?
13. Do you test with the Iteration builds?
14. When you do reviews, what is the information you provide into the team?
15. So, you do one to one pairing with an engineer? Regularly during iteration?
16. Are working closely with other departments such as QA, documentation etc?
17. Have seen any difference with your DAD project team on the understanding of personas or on user study? Any major improvements?
18. Does agile provide benefits to UX? Does agile cater to the needs of UX?
19. The final product through DAD approach, does the standard live up to your expectation as UX?
20. Still 'us vs them' culture, does an engineer think differently?
21. What about product backlog? Does it serve your needs and provide good understanding? Your work is incorporated there?
22. As PO, do you get visibility when you look at product backlog?
23. As PO, the estimates, are they as close to what was estimated in delivery?
24. When you get product backlog, is it as close as the estimates.
25. You have some certain criteria the team must follow to put things into product backlog? What are they?
26. Previously you were using use cases, but now user stories. Why?

A.3.5 DAD adoption & Product management, Director of Software Engineering–1st interview, 28th March 2017

1. Was moving to DAD & cloud deployment a business-driven decision?
2. How much planning was done and how much time it took to a successful transition to cloud?
3. How was this decision seen by the engineering team?
4. Did they have any input regarding the decision making?
5. So, did you have any changes to the functional units on the business and engineering side?
6. Did the organization evaluate the software engineering capabilities before moving to cloud?

7. How long did it take for the team to adapt to DAD approach and to cloud application development?
8. The product management team where do they fit in now, is it still based in USA?
9. So you got a co-located project management person here? So you're finding a lot of design decisions are being made spontaneously, don't have to wait?
10. Were there any major changes to the roles and responsibilities with engineering when you moved to DAD?
11. So there's a lot of informal roles and the formal roles for the software engineers ?
12. Have you encountered any major challenges or issues with having this kind of setup, where you have got so many leaders, clash of ideas?
13. Changes to the role of product development manager?
14. How did you overcome this challenge?
15. What about in terms of putting a QA person in the team, is it the way it used to be?
16. So, you also have embedded documentation into the DAD project teams–how is it going?
17. So, what is your development capacity at the moment in terms of the number of engineers you have here, USA and India?
18. Where does this usability/UX person fit in?
19. So roughly, what will be the capacity with UX team? 3-4?
20. What is the duration of your DAD programs and projects like now, compared to the hybrid approach you had ?
21. What about market releases?
22. There are major differences in the way you are doing things now compared to on-premise application development under the hybrid approach?
23. In each DAD project team, what will be the number of engineers?
24. Do you shift people from one DAD team to another?
25. So now with DAD approach you just moved away from the squad concept?
26. What about the role of DevOps practice now?
27. What about new tools and technologies, how do you decide what to adopt?
28. Back to DevOps, normally operations would want a stable environment, and developers want to play with things, how do you handle this?
29. How far away are you from giving a job title as DevOps for software engineers?
30. Where there any major problems/ challenges with the level of technical skills and product knowledge when you moved to cloud application development including able to adopt DAD team structure and roles?

31. So you needed a technical champion when it comes to DAD adoption?

32. Without a hybrid agile approach, if you were stuck with RUP, would you be able to successfully transition to DAD approach?

A.3.6 Program manager, Director of Software Engineering, 2nd Interview–30th March 2017

1. DAD approach you would also have to align your role and responsibilities according to this approach to DAD?
2. So, in terms of your responsibilities (DAD) as program manager now you are planning development capacity?
3. As program manager, You will be responsible to ensure project is distributed among the engineering teams here, USA and India?
4. Do analysis and design work at program level?
5. Do you also provide domain support for projects under the program?
6. What are the key things you probably want to keep an eye on for projects in a program?
7. The DAD methodology, who identified it?
8. DAD method does it help integration and collaboration between engineering and product management?
9. What change has Product management incorporated as a result of DAD adoption?
10. What about during the project, would you have a significant change during projects?
11. How do you as a program assign projects to teams, what are the criteria for that?
12. Are your DAD teams based on functionality of your product? Why?
13. How many local DAD project teams and how many offshore DAD projects teams come under your program?
14. Are the culture of each DAD project team the same or different? Do you emphasize core culture?
15. The definition of done, what are the things in it?
16. Your hiring process, you adapt it with DAD to match your approach now?
17. In inception, to secure funding, once the team spend time, will they need to come up with some deliverable to convince stakeholders?
18. Do you still use business cases, Product Managers do them?
19. At program level who does high-level estimates?
20. Explain the role of enterprise architects at program level and project level?
21. How does enterprise architect role differ from architect owner role in projects?

22. Which software engineer level is an ideal level for enterprise architect role and who can work in architect owner role?
23. The DAD philosophy, people first, who are the people here? Stakeholders? Team members?
24. Can you identify your practices at program level including roles?
25. What are some of the agile method practices that are common to all the teams under your program
26. Work item list, how does it differ from product backlog?.
27. When providing user story estimates at program level do you also provide story points?
28. So, estimation is all done collaboratively at program level?
29. Is work item list business focused or engineering focused?
30. DevOps practices, can you elaborate on them?
31. Your construction phase, briefly talk about those iteration releases?
32. Duration of inception phase?
33. Do you think a lot of work in inception phase should be done before project starts at program?
34. In the funneling process at program level, who are the key people involved?

A.3.7 Architecture Owner, Principal Engineer–30th March 2017

1. In terms of your role, how has it changed in DAD?
2. You are kind of doing work with funneling the features at program level?
3. When you are in inception phase, do you have any architect solution for the entire project?
4. What is your involvement during construction and transition phases?
5. In inception phase, are you required to create certainty for program and project?
6. Feedback from team, work closely with team to come up with something? Collective decision?
7. How do you choose the tools and open source software that you use for the development of cloud?
8. What cloud platform do you use?
9. How much collaboration do you have with operations?
10. Do you make decisions on the development infrastructure here?
11. When you sort out your architecture, what are the things that you have to consider in terms of supporting other systems?
12. Is your architecture built on shared resources? Multi-tenant?
13. When you do the architecture, do you have to consider your data center a lot?

14. How you ensure security in multi-tenancy?
15. Multi instance, have you considered that as well? One application one tenant.
16. How many architecture layers do you have?
17. Some of the best practices that you use for your architecture design?
18. Are clients able to personalize? Flexibility?
19. What about self-service?
20. In terms of integration with other application?
21. Operational performance, is that an absolute necessity in your product?
22. Security is that another critical element you have to ensure?
23. With your DAD adoption, how much discussion was done with you guys before adopting it?
24. What about training with Ambler? Coaching for DAD?
25. Does the process factor in the transfer of experience?
26. At the inception phase, you don't have product backlog, you have work item stack. What is your view?
27. Your View on the planning tool Jira, does it work well?

A.3.8 Quality Assurance Manager–29th March 2017

1. Changes in roles and responsibilities when you moved to DAD?
2. Compared to before, which approach do you prefer?
3. To-do list, what tasks are part of the list with the DAD approach?
4. In the non-functional test, what things do you test for?
5. Anything else under non-functional test?
6. You do usability as well?
7. What are the other things when it comes to reliability testing?
8. What about performance, is it different from reliability testing?
9. What about the automation? Are there any tests you do manually?
10. The infrastructure features for automation are they sufficient and fit well with the DAD approach?
11. This is what is different from what you used to do before DAD approach?
12. Operation want a stable environment; do you get the chance to test the operational environment?
13. You have various regulations Worldwide? Who is responsible to identify them?

14. Do you have an overall QA plan?
15. Team in Bangalore is doing more manual testing?
16. Who does core reviews now?
17. Regarding the escaped defects that you mentioned, compared to on-premises, which is more severe?
18. Since adoption of DAD, are there major defects or bugs being found after releases?
19. The roles that you had before, how it was and how it is now? Are you more of an agile QA now? Providing more support, training, and coaching? Rather than sitting down and testing?
20. QA challenges with DAD approach?
21. In terms of up scaling your QA based on DAD approach, was it a challenge?
22. Do you have someone who is the technical champion who looks for testing technologies?
23. What are the levels in QA and how do they fit in a Dad setup?
24. So, you have a lot of levels in that sense?
25. Shifting to DAD, do you think QA will require more technical skill sets?
26. Does DAD require mostly test automation?
27. What about Virtualization?
28. Testing 75 different systems, is still being done with your DAD approach?
29. What tools do you use to get testing data?
30. What other thing do you feed back to the development team that comes out of monitoring?